3-1-2012

Characterization of speaker recognition in noisy channels

Robert Ghilduta

Characterization of Speaker Recognition in Noisy Channels

by

Robert Ghilduta

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Engineering

Supervised by

Professor Dr. Andres Kwasinski
Department of Computer Engineering
Rochester Institute of Technology
Rochester, NY
March, 2012

Approved by:

_____ Date: _____
Dr. Andres Kwasinski
Primary Advisor – R.I.T. Dept. of Computer Engineering

_____ Date: _____
Dr. Roy Melton
Secondary Advisor – R.I.T. Dept. of Computer Engineering

_____ Date: _____
Dr. Muhammad Shaaban
Secondary Advisor – R.I.T. Dept. of Computer Engineering

Acknowledgements

I would like to thank my thesis advisors, Dr. Andres Kwasinski, Dr. Roy Melton, and Dr. Muhammad Shaaban for their suggestions, advice, and support.
I would also like to thank all of my friends whose support and feedback has been very valued throughout this thesis.

Contents

List of Figures

List of Tables

Abstract

Speaker recognition is a frequently overlooked form of biometric security. Text-independent speaker identification is used by financial services, forensic experts, and human computer interaction developers to extract information that is transmitted along with a spoken message such as identity, gender, age, emotional state, etc. of a speaker.

Speech features are classified as either low-level or high-level characteristics. High-level speech features are associated with syntax, dialect, and the overall meaning of a spoken message. In contrast, low-level features such as pitch, and phonemic spectra are associated much more with the physiology of the human vocal tract. It is these low-level features that are also the easiest and least computationally intensive characteristics of speech to extract. Once extracted, modern speaker recognition systems attempt to fit these features best to statistical classification models. One such widely used model is the Gaussian Mixture Model (GMM).

The current standard of testing of speaker recognition systems is standardized by NIST in the often updated NIST Speaker Recognition Evaluation (NIST-SRE) standard. The results measured by the tests outlined in the standard are ultimately presented as Detection Error Tradeoff (DET) curves and detection cost function scores.

A new method of measuring the effects of channel impediments on the quality of identifications made by Gaussian Mixture Model based speaker recognition systems will be presented in this thesis. With the exception of the NIST-SRE, no standardized or extensive testing of speaker recognition systems in noisy channels has been conducted. Thorough testing of speaker recognition systems will be conducted in channel model simulators. Additionally, the NIST-SRE error metric will be evaluated against a new proposed metric for gauging the performance and improvements of speaker recognition systems.

Chapter 1

Introduction

Speaker recognition is becoming more ubiquitous as reliance on biometrics for security and convenience increases. Speaker recognition systems have been used in both speaker verification and speaker identification applications. Speaker verification is primarily used to verify the claimed identity of a speaker. As an example, financial institutions may prompt customers to speak a personalized verification phrase such as their personal identification number (PIN). The speech sample is analyzed and compared against a speaker model associated with the account owner. Speaker models are trained during an enrollment session, which generally consists of prompting the new user to repeat an acoustically diverse phrase when their account is created. In this case, generally text-dependent speaker recognition is used as a closed-set biometric security tool granting access to a protected account. Speaker identification is concerned with identifying a speaker from a pool of possible speakers; such implementations are generally text-independent. As an example, a home theater console can identify the person using the set and offer the person a customized entertainment experience.

This thesis will provide the necessary background knowledge about speaker recognition systems in chapter 1. The noise and channel models that will be investigated are

detailed in chapter 2. The primary contributions of the thesis are then presented. The justification for the use of k-means initialized GMMs is presented in chapter 3. The proposed metric for ascertaining the performance of speaker recognition systems is presented in chapter 4. Ultimately, chapter 5 will present thorough testing of channel impediments on the performance of speaker recognition systems as gauged by the proposed metric.

1.1 Theory

A speaker recognition system is comprised of many subsystems as is shown in Figure 1.1. Signal processing and feature extraction blocks are responsible for conditioning a signal and extracting low-level features. Speech features can be low-level, (e.g., the spectra of the sound of someone's voice, pitch, emotional state, gender, etc.), or high-level features (e.g., vocabulary, syntax, dialect, style, etc) [2]. Low-level features extracted from short-term spectral analysis are the most widely used kind in text-independent speaker recognition systems. The use of extracted low-level features has proven to be reliable and effective, in addition to being computationally inexpensive. After channel dependent signal conditioning is performed, the front-end analysis extracts spectral features.

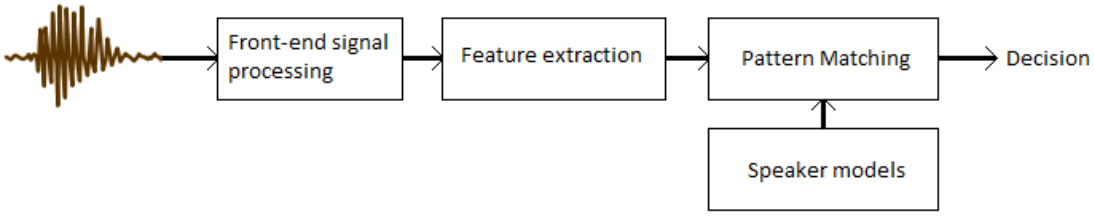

Figure 1.1: Block diagram of a typical speaker recognition system

Low-level features are best explained in terms of the physiology features of the human vocal tract, Figure 1.2. The larynx and the vibrating vocal folds are the primary sound sources found in the vocal tract. The vocal folds are composed of five layers of

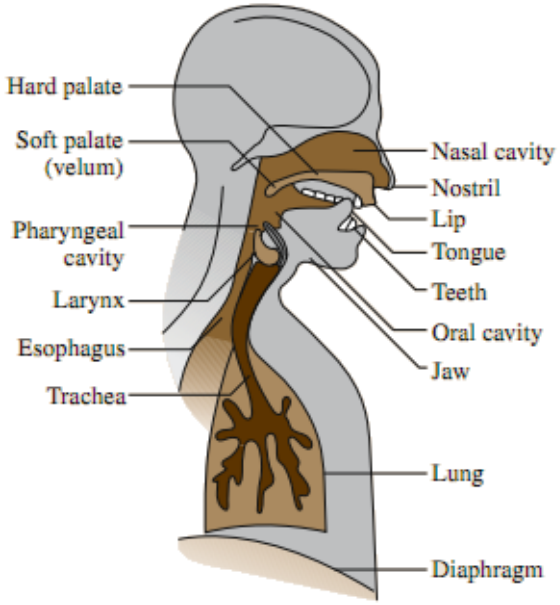

Figure 1.2: Physiology of the human vocal tract

tissue that are under conscious control. As air is forced outward from the lungs, pressure causes the closed vocal folds to separate. The escaping air leaves the focal folds and enters the larynx and pharynx. As this occurs the vocal folds rhythmically snap shut, and sound is created. The fundamental frequency of a speaker's speech is attributed to this rhythmic vibration [3]. The fundamental frequency is modulated by the remaining part of upper vocal tract, primarily by the shape of the nasal tract. The combined effects of the vocal folds and the vocal tract give each person their own unique characteristic voice fingerprint. The fundamental frequency of speech is invaluable to speech coders for better compression capabilities and to speaker recognition systems for better speaker differentiation. Additionally, information such as identity, health, age, and emotional state directly affect the vocal tract.

Modern speaker recognition systems exploit knowledge about the physiology of the vocal tract by using digital signal processing to extract low-level from speech. Some im-

plementations such as Reynold's system, segment audio samples into 20 ms long frames at a repeating 10 ms rate. Mel-frequency cepstral coefficients (MFCC), a widely used spectral representation of human speech based on psychoacoustics, the human perception of speech, are applied to the sample frames [4]. Mel frequency cepstral coefficients are the discrete cosine transform (DCT) coefficients of the log energy of triangular filters applied to the discrete Fourier transform (DFT) coefficients of a windowed audio frame. Typical window functions include Hamming and Hanning functions [5]. In the MFCC representation, cepstral bins are uniformly (linearly) spaced from 0 to 1 KHz, and logarithmically above 1 KHz. The per-frame sets of extracted MFCCs are defined as the feature vectors.

1.1.1 Gaussian mixture model

There is no single or unique feature that can be used to identify a speaker, therefore extracted features are passed on to a pattern matching block, which usually consists of a Gaussian mixture model (GMM). GMMs are a proven statistical tool by which the low-level features of human voices can be modeled [6]. Gaussian mixture models do not require a speaker recognition system to have prior knowledge of text or other low-level parameters of a given speaker. As such, Gaussian mixture models have become the most widely used text-independent speaker models, trumping vector quantization (VQ) [7], artificial neural networks [8], hidden Markov models (HMM) [9], and for the time being support vector machines (SVM) [10].

The underlying assumption made by speaker recognition systems is that audio samples that come from the same speaker, are sufficiently long and diverse enough phonetically will have highly similar MFCC distributions. Conversely, audio samples from distinct speakers will have largely different MFCC distributions. MFCC distributions, the feature vector sets, are histograms of each MFCCs's values across an entire audio sample. In speaker recognition systems, Gaussian mixture models provide the statistical tool by

which single MFCC feature vector sets can be quantitatively gauged as belonging to a specific speaker's model.

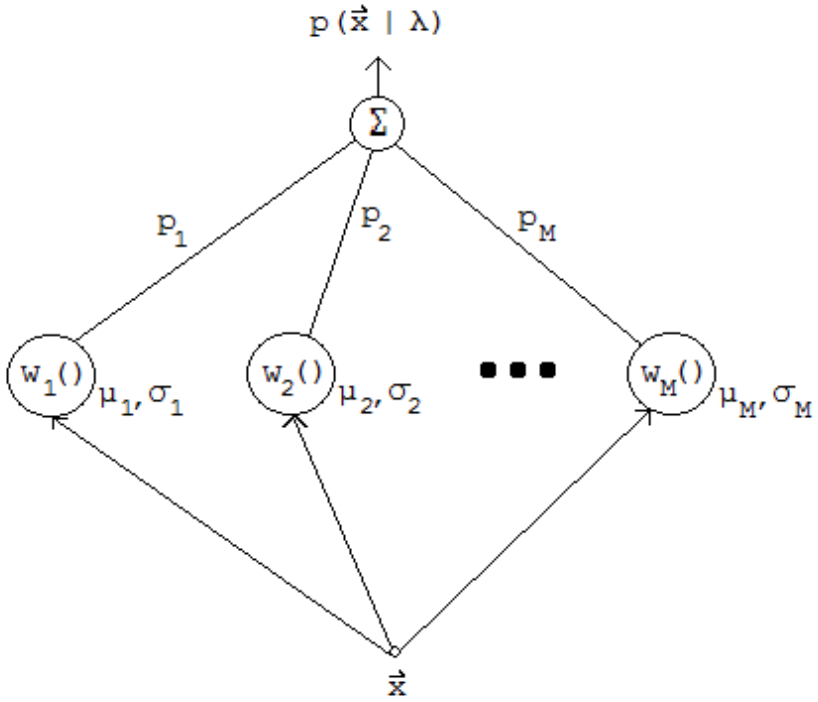

Figure 1.3: A set of independent weighted Gaussian distributions comprising a Gaussian Mixture Model

A trained speaker model can be represented by a Gaussian mixture model, denoted by $\vec{\lambda}$, consisting of three vectors of parameters: means ($\vec{\mu}$), variances ($\vec{\sigma}$), and weights ($\vec{w_m}$). Hence each MFCC is represented by a weighted sum of underlying Gaussian distributions, as is shown in Figure 1.3. Before a speaker model is trained, its initial parameters must be carefully chosen in order to help the parameter refining algorithm converge. Modern speaker recognition systems compute means and variances from randomly selected MFCC frames belonging to the audio sample being analyzed. Given a set of feature vectors **X** an iterative expectation-maximization (EM) algorithm computes the maximum-likelihood (ML) estimates for the model by a two-step process. At each iteration the pos-

terior probability of feature $\vec{x}_t$ belonging to $p_m(x_t \mid \lambda_m)$ is calculated Eq. 1.1.1.1. Secondly, each of the Gaussian parameters of λ is updated [4]. In the iterative process, mixture weights are evaluated by Eq. 1.1.1.2, and the means and variances of each Gaussian distribution are evaluated by Eq. 1.1.1.3, and Eq. 1.1.1.4, respectively.

$$P(m \mid x_t, \lambda) = \frac{w_m p_m(x_t \mid \lambda_m)}{p(x_t \mid \lambda)} \tag{1.1.1.1}$$

$$\overline{w}_m = \frac{1}{T} \sum_{t=1}^{T} p(m \mid \vec{x}_t, \lambda) \tag{1.1.1.2}$$

$$\vec{\overline{\mu}}_m = \frac{\sum_{t=1}^{T} p(m \mid \vec{x}_t, \lambda) \vec{x}_t}{\sum_{t=1}^{T} p(m \mid \vec{x}_t, \lambda)} \tag{1.1.1.3}$$

$$\overline{\sigma}_m^2 = \frac{\sum_{t=1}^{T} p(m \mid \vec{x}_t, \lambda) x_t x_t^T}{\sum_{t=1}^{T} p(m \mid \vec{x}_t, \lambda)} - \overline{\mu}_i \overline{\mu}_i^T \tag{1.1.1.4}$$

The EM algorithm guarantees convergence of the estimated parameters. Once trained, the λ model is a statistical representation of a speaker's voice. In the case of speaker identification, the trained λ parameters are used to determine if the set of feature vectors Y from an audio sample were produced by the same speaker. Ultimately a similarity score is given by the average log-likelihood (Eq. 1.1.1.5).

$$s(Y \mid \lambda) = \frac{1}{T} \sum_{t=1}^{T} log[p(y_t \mid \lambda)] \tag{1.1.1.5}$$

The likelihood score (shown in Eq. 1.1.1.6) is a Gaussian test of whether a feature belongs to a model, where the null hypothesis H_0 represented by λ_{hyp} gives the probability of feature X belonging to the model.

$$\text{likelihood ratio score} = \frac{p(\mathbf{X} \mid \lambda_{hyp})}{p(\mathbf{X} \mid \lambda_{\overline{hyp}})} \tag{1.1.1.6}$$

Generally the logarithm of the likelihood ratio score, known as the log-likelihood ratio (LLR) shown in Eq 1.1.1.7, is used as the soft output which is then interpreted as either "true" or "false" and tallied in to a final test metric.

$$\Lambda(\mathbf{X}) = \log p(\mathbf{X} \mid \lambda_{hyp}) - \log p(\mathbf{X} \mid \lambda_{\overline{hyp}}) \tag{1.1.1.7}$$

1.1.2 Existing Work

Various speaker recognition systems have been explored in the literature. Gaussian mixture models have been developed in [4] [11]. Support vector machine based models have been discussed in [10]. In addition to implementation methodologies, standardized testing has been encountered in [12]. The NIST-SRE, however does not provide guidelines for testing speaker recognition systems in noisy channels. Vocoders have been investigated in [13], however only a single bitrate of a GSM codec was investigated. [14] provides insight into the degradation of phonemes in audio samples that pass through CELP codecs. The effects of the analog based Public Switched Telephone Networks communication channels were explored by Reynolds in [15].

The development of a speaker recognition system is not unique in that it is driven by requirements such as use cases, characteristics of training and testing speech corpora, error rate thresholds, computational power limits of target host device, etc. In the literature, Gaussian mixture models are widely used in embedded applications primarily due to the efficiency of expectation-maximization algorithms [16] [17]. Computational efficiency is very beneficial in applications that must use their own computational power to train their models. The run-time computational complexity of training a system is caused by the inherent computational expense of adapting a model's parameters through an iterative process that requires the model to process feature sets for each iteration.

Once a model is trained, the computational power of a single one iteration of an expectation-maximization algorithm step is needed to identify a speaker. Generally, this holds true for most speaker recognition systems regardless of the underlying model.

Alternatives that have been encountered but will not be explored include outdated statistical classifiers such as vector quantization (VQ) [7]. Although not universally deprecated, VQ classifiers have been surpassed by GMMs in the text-independent speaker recognition systems [4]. Hidden Markov models (HMMs) prove to be very useful in text-dependent applications. However, they only offer substantial benefits in text-independent applications [9]. HMMs will not be investigated due to their completely different assumptions from GMMs on the an underlying model of speech. The increase in popularity of support vector machines (SVMs) [18] has lead to the development of more efficient techniques. Compared to GMMs, SVMs have only recenlty attracted serious attention from the speaker recognition community; thus many implementation methods are not as mature.

1.1.2.1 Testing

The observed testing methodologies applied to speaker recognition systems have ranged from simple error rate tests [4] to thorough NIST-SRE compliant testing [12]. Testing of speaker recognition systems in environments where the audio quality of the training and testing audio samples differ has been investigated in [19].

In the aforementioned literature, speech corpora (e.g., KING, SPIDRE, TIMIT, etc.), which consist of several minute long phone conversations, are used as audio samples for training and testing the speaker recognition systems. The NIST-SRE guidelines, that are followed by other works as well, suggest varying certain system parameters including the length of the training audio, the number of underlying Gaussian distributions in a GMM, and the selection of Mel cepstra. However, most speaker recognition tests are carried by varying audio sample lengths and microphone setups.

The most formalized testing method is outlined by the NIST-SRE standard which details four different training conditions (10-sec, core, 8conv, and 8summed) and three test conditions (10-sec, core, summed). The 10-sec and core specifications require single conversation recordings lasting exactly 10 seconds and at most 15 minutes, respectively. 8conv, and 8summed are similar to the previous two specifications with the exception that audio samples come from 8 different conversations by the same speaker. Additionally, 8summed requires a speaker recognition system capable of coping with audio samples that include both sides of a phone conversation, specifically the non-target speaker. The NIST-SRE only requires the core-core training-test condition to be completed.

$$\begin{aligned} C_{Det} = C_{Miss} \times P_{Miss|Target} \times P_{Target} \\ + C_{FalseAlarm} \times P_{FalseAlarm|NonTarget} \times (1 - P_{Target}) \end{aligned} \tag{1.1.2.1}$$

The NIST-SRE also specifies the detection cost function metric given in (1.1.2.1).The metric seeks to penalize the speaker recognition system whenever it fails to identify a speaker correctly. To account for test scenarios where not all speakers have an equal number of test audio samples, the metric corrects for the distribution skew by taking the probability (P_{Target}) of speakers into account. The probability (P_{Target}) is the ratio of audio samples belonging to speaker $Target$ in the audio collection. Misses are defined as incorrectly identifying $Target$ (the speaker whose audio sample is under test) when they in fact are the speaker. Conversely, false positives are defined as incorrectly identifying $Target$ as the speaker when they in fact are not. In test scenarios where every speaker has the same probability of appearing, misses and false positives are essentially the same. Additionally, C_{Miss} and $C_{FalseAlarm}$ are experimentally derived constants that allow for detection error trade off (DET) curves [20]. DET curves of similar implementations tend to be parallel to each other.

Ad-hoc, or not standardized, testing methods generally tend to rely solely on error rates as a means of ascertaining performance of a system when applied to a speech corpora. A difference between the testing of GMM and SVM based implementations has been observed [10]. While GMMs are capable of directly computing a scalar log-likelihood, SVM classification outputs have to be normalized to a term frequency log-likelihood ratio (TFLLR) [18]. This additional step adds a layer of abstraction that complicates the implementation of advanced error measurement.

Due to the popularity of cellular phones, testing would be remiss without investigating the effects of speech encoders used in telecommunications on speaker recognition systems. A thorough survery of the effects of modern voice encoders such as AMR, EVRC, and Speex will be conducted to ascertain the effects of compressed audio channels on speaker recognition systems. The experiments will distinguish itself by considering the encoders that combined account for the overwhelming majority of encoded conversa-

tions and their effects on the more popular implementation (text-independent) of speaker recognition systems.

1.1.3 Proposed development

Log-likelihood scores and NIST-SRE results provide a means to benchmark a speaker recognition system. They however do not provide a means of gaining intuitive knowledge of how a system reacts to variations in training and test conditions. Log-likelihood scores are scalar values and can be interpreted only relative to other log-likelihood scores. In speaker verification the speaker with the highest score is considered to be the identified target. For testing purposes a higher discrepancy between speakers will yield better test results. In training and development, successively higher scores could be caused by better trained models, front-end processing techniques, or expectation-maximization methods.

Although the NIST-SRE error rate, the detection cost function score, of a system is an adequate metric of its working performance, it may not be the best metric for parameter tuning purposes. In case a speaker recognition system is tuned for a specific operating environment, using an error metric that is better able to ascertain changes within the statistical classifier might prove more useful to an iterative improvement method (i.e. a genetic algorithm) than NIST-SRE error rates alone. By creating a metric that works with the classifier's output directly, a deeper understanding of changes to GMM parameters can be achieved leading to possibly faster and better convergences in the case of genetic algorithms. The metric will be tested and compared against the NIST-SRE error rate for use as a metric in various forms of channel impediments. A comprehensive test of the effects of channel impediments against speaker recognition systems will be conducted. Additionally, the proposed metric will be used as an error metric for algorithms that tune mixture parameters based on the channel impediments the system is encountering.

1.1.3.1 Implementation

A Gaussian mixture model based text-independent speaker recognition will be developed according to modern practices [6]. Feature extraction will be performed by taking the previous described Mel-frequency cepstrum transform of audio samples. A k-means initialized Gaussian mixture model will be used as the principal statistical classifier. K-mean centroid GMM initialization will be proposed and examined further in this thesis. Ultimately, speaker verification will be conducted using log-likelihood ratio scores.

1.1.3.2 Proposed Metric

The thesis will describe the development of an error metric capable of quantifying improvements to a system as a time-domain vector set from the perspective of the classifier. Such a metric will allow a developer of a speaker recognition system to better understand the effects of changes to the performance of the system in a time-domain feature metric. The metric can be used to gauge the susceptibility of an implementation to channel impediments such as noise, speech encoders, and recording sources. The implementation of the metric will be compared with expectations from psychoacoustic studies.

Iterative improvement methods such as genetic algorithms require metric functions to determine the viability of changes to a system. As such, the new metric will be compared against the NIST-SRE error rate as the metric function for use in a genetic algorithm. The new metric will favor improvements that increase correlation peaks while maintaining correlation averages, thus possibly leading to faster and better convergences than the NIST-SRE error rate.

1.1.3.3 Analysis of GMMs in Noisy Channels

The speaker recognition system will be subjected to various forms of channel impediments to benchmark its performance. Since most speaker verification applications are generally conducted over telephone conversations [21], an analysis of speech encoder effects on speaker-identification systems will be conducted. Decoding compressed audio produces artifacts that can not be easily estimated by any noise model or linear filter. As such, conclusions and testing methodologies derived from previous rounds of analysis will be used to ascertain the performance of the speaker recognition system when its speech corpora are encoded using GSM's AMR, CDMA's EVRC, and Speex vocoders. Testing will attempt to establish the effects of low bitrates as well as mismatched training and testing situations.

Chapter 2

Noise and Channel Models

Various noise and channel models have been selected to gain insight into the effects of real world channels on speaker recognition systems. Different forms of noise such as white, pink, and Brownian noises were selected as suitable models for naturally occurring phenomenon that speaker recognition system channels frequently encounter. Generally, sources of noise are encountered by speaker recognition systems during the acquisition of audio samples. In addition to forms of noise, channels models such as Additive White Gaussian Noise, multipath, voice coders are also investigated.

Channel models lend themselves to modeling the transmission of audio in wired, wireless, analog, digital, and coded and unencoded channels. Noise and channel models can occur in any combination and permutation but it is the intent of this thesis to cover widely encountered situations.

2.1 Noise Models

2.1.1 White Noise

One of the elementary forms of noise distributions is white noise. In its simplest case, white noise is characterized by a constant power spectral density, which is a function of power (dB) per bandwidth (Hz), across the entire spectrum. The short window discrete-time spectrogram and Fourier transform of typical white noise are given in 2.1.

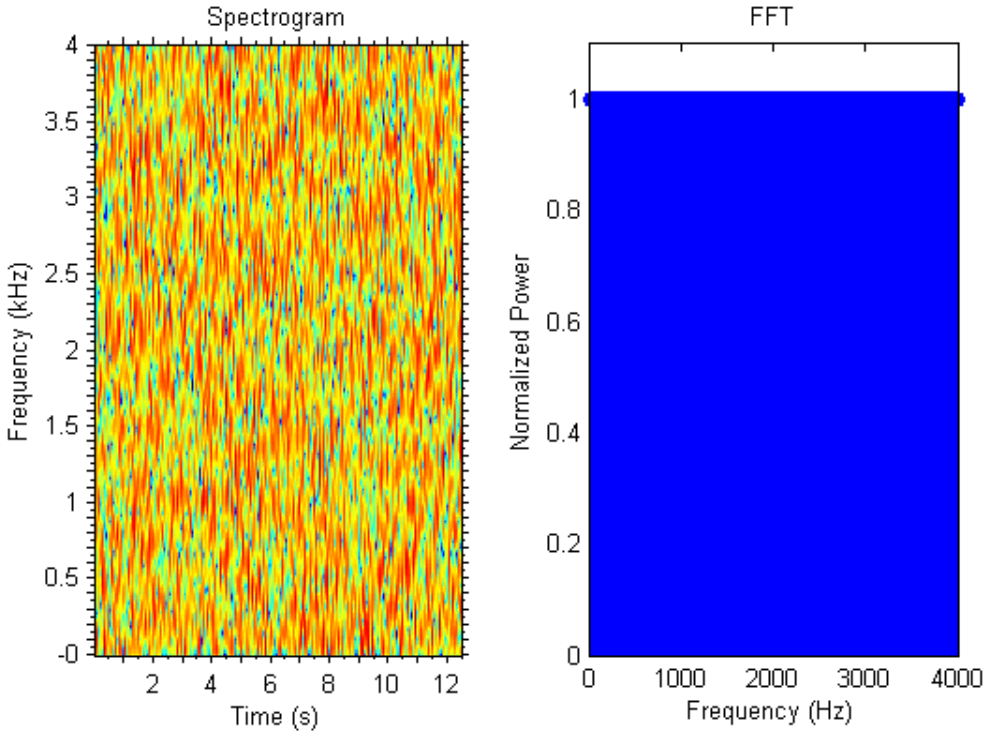

Figure 2.1: Spectrogram (left) and FFT (right) of White Noise

Another important factor in the characterization of noise is autocorrelation. The real-valued discrete-time autocorrelation of a random process, denoted by X, is given by the autocorrelation formula Eq. 2.1.1.1. The discrete-time difference between indexes s and t is the period of the correlation test. Variables μ and σ describe the mean and variance

of process X, respectively. A correlation is bounded to [-1, 1], where -1 signifies a perfect anti-correlation, 1 signifies a perfect correlation, and 0 means uncorrelated. The expected discrete-time correlation of white noise with any values for s and t is equal to 0.

$$R(s,t) = \frac{E[(X_t - \mu_t) * (X_s - \mu_s)]}{\sigma_t \sigma_s} \tag{2.1.1.1}$$

Lacking any sort of correlation implies that any two samples of a random process are independent of each other. Linear predictive coding (LPC), which is used by voice encoders for compression purposes is heavily affected by white noise since the underlying excitation prediction algorithms fail.

White noise was selected as a suitable model for various forms of noise encountered in applicable real world scenarios. Physical phenomena such as thermal noise in electronic components, interfering static electrical discharge, wind blowing into a microphone, and non-specific faded background city noise are very well modeled by white noise at various SNR levels.

2.1.2 Additive White Gaussian

Speaker recognition systems may be deployed in systems where a sampled audio stream is sent wirelessly using an analog modulation scheme. Common handheld transceivers such as "walkie talkie" handsets are perfect examples.

Analog modulation schemes such as amplitude modulation (AM) and pulse-amplitude modulation (PAM) are highly susceptible to noise. An analog amplitude modulation transceiver system is unable to distinguish noise regardless of its origin from the original modulated signal and it will introduce the same noise in to the demodulated signal.

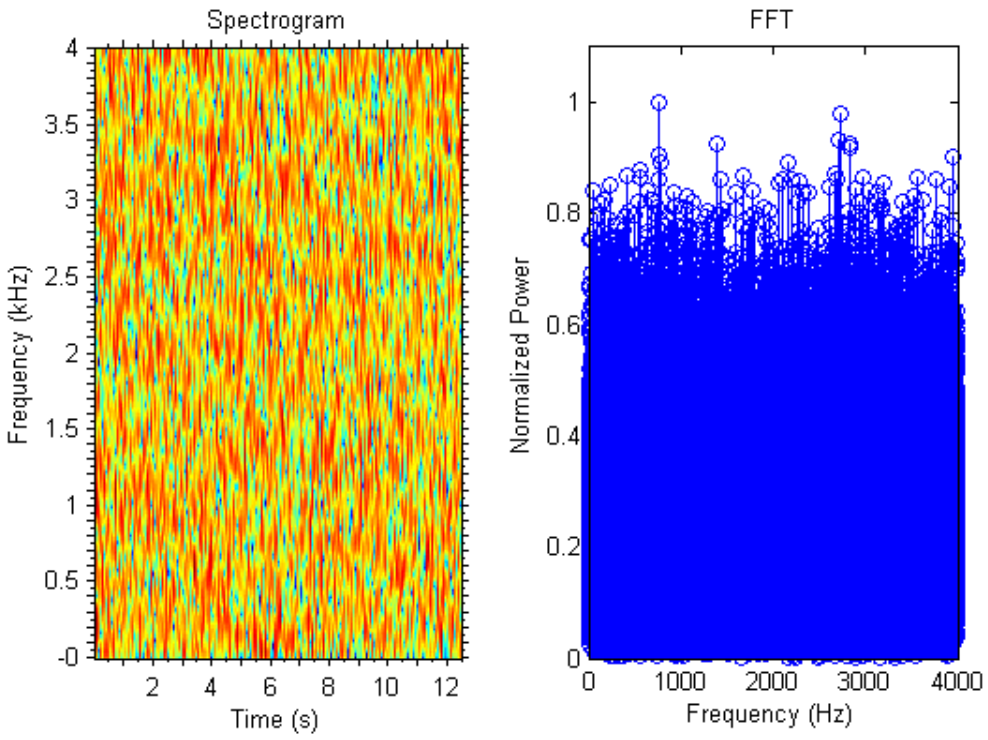

Figure 2.2: Spectrogram (left) and FFT (right) of Additive White Gaussian Noise

Additive white Gaussian noise (AWGN) serves to model background noise encountered by the wireless transmission channel. Other prominent effects such as fading, and multipath are not considered. During preliminary testing, echoes created by wireless multipath were found to have a negligible effect on the quality of voice communications in analog modulation schemes. By considering the received power, thus the range, of analog modulation based two-way communication handsets, it is improbable for RF multipath to create echoes that are delayed by more than even a tenth of the period of the highest frequency component of the transmitted audio signals.

2.1.3 Pink Noise

Unlike white noise, pink noise does not have a constant spectral power density. As a power noise (also known as $1/f$ noise), pink noise has a frequency dependent spec-

tral power density. As the name indicates, the power spectral density of the random process is inversely proportional with frequency, as shown in Figure 2.3. Pink noise serves as a very good model for power regulation noise that may exist on poorly designed or tightly packed circuit boards.

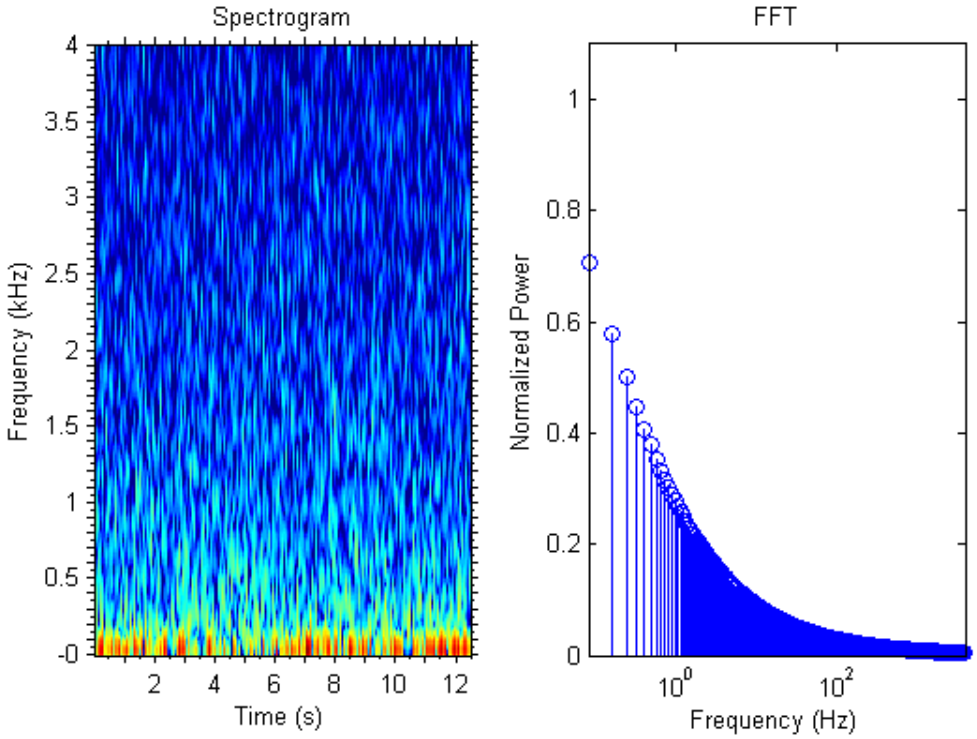

Figure 2.3: Spectrogram (left) and FFT (right) of Pink Noise

Modern circuit boards use many various voltage standards for digital electronics that come from an equally abundant number of current supplies (e.g. lithium batteries, NiMH batteries, solar panels, USB and wall chargers). Board designers often times use switch mode power supplies (SMPS) as a means of stepping voltage up or down over very large voltage ranges for efficiency purposes, and use linear power regulators to smooth out the ripple from SMPSs. Switch mode power supplies work by maintaining the output voltage rail at a reference voltage by varying the duty cycle of a electromagnet in the switch mode power supply. Because of the low frequency, switching speeds and intermodulation

products that occur SMPSs generate a lot of noise in the low frequencies, which coincidentally are also the hardest to filter properly [22]. Pink noise is an appropriate model for the noise an analog component such as a microphone could potentially intercept from a poorly designed or poorly filtered power regulation system. Switch mode power supplies are present in cellular phones as well as in landline telephones, therefore their inclusion in testing is necessary. Pink and flicker noise are specific cases of $1/f$ power noise known as $1/f^1$. The spectral power density of the random process decreases 3dB per octave.

2.1.4 Brownian Noise

With a spectral power density that is inversely proportional to second power of frequency, Brownian noise is a form of power noise. Unlike pink noise, Brownian noise's spectral power density decreases 6dB per octave; thus it is also known as $1/f^2$ noise. Brownian noise was a selected as an alternative model for fast switching (high frequency) switch mode power supplies. Smaller inductors require faster switching times, which cause most of the harmonics and intermodulation spectra to wind up several octaves beyond the cut off frequency for analog microphone front-ends. This characteristic of fast switching SMPSs makes Brownian noise a better model for the electronics that use them.

2.2 Channel Models

2.2.1 Multipath

Multipath is a complex phenomenon that encompasses the effects of reflection, diffraction, and scattering of propagating waves. As electro-magnetic, and acoustic waves propagate through space they encounter obstacles. The manifestation of multipath depends solely on the properties of the wave, the medium through which it propagates,

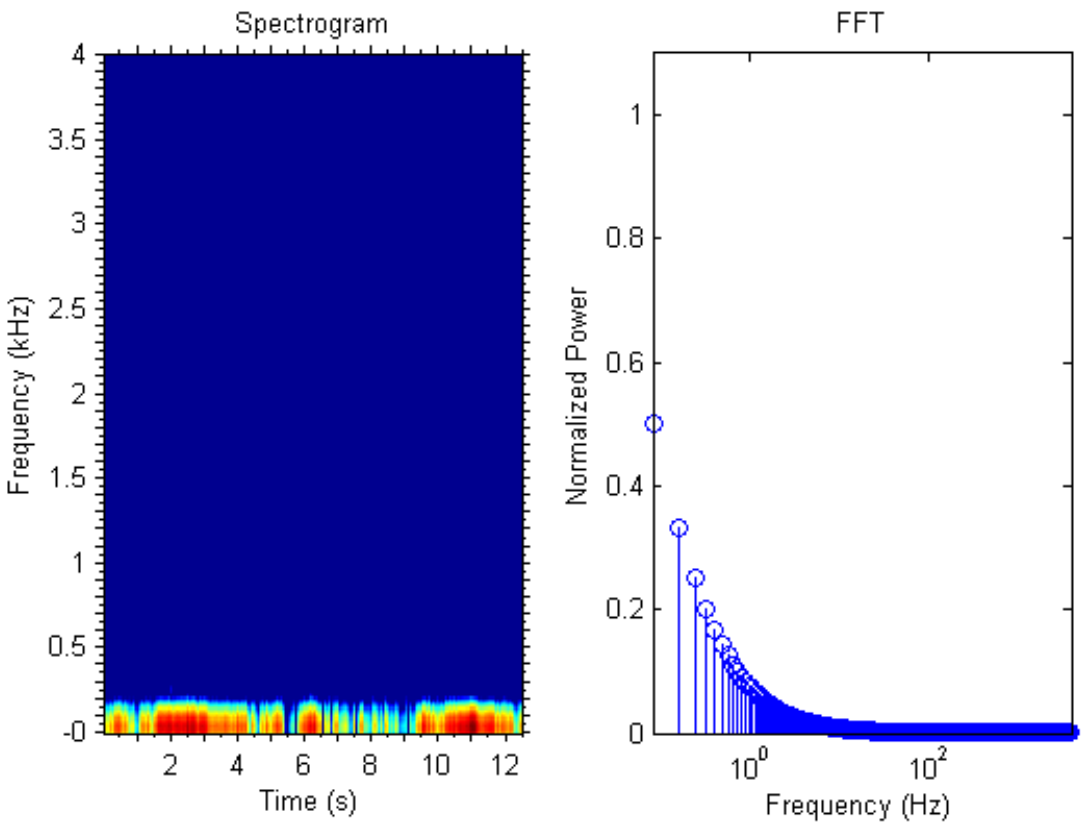

Figure 2.4: Spectrogram (left) and FFT (right) of Brownian Noise

and the properties of the materials and geometries the obstacles consist of.

Commonly, multipath is viewed as a series of attenuated reverberations caused by obstacles creating reflection, as shown in Figure 2.5. Assuming there is a direct line of sight between a wave source and a receiver, some of the waves propagating from the source will reach the receiver directly, thus taking the shortest and least attenuated path. Other waves will stray off in other directions and reflect off obstacles that will, at the very least, attenuate them and change their direction. Some of the reflected waves will then propagate toward the receiver with varying altered characteristics and reach the receiver with time delays.

Multipath channels can be accurately modeled by a series of impulses representing phase shifts and time delays, as shown in Figure 2.6. One way to model multipath is

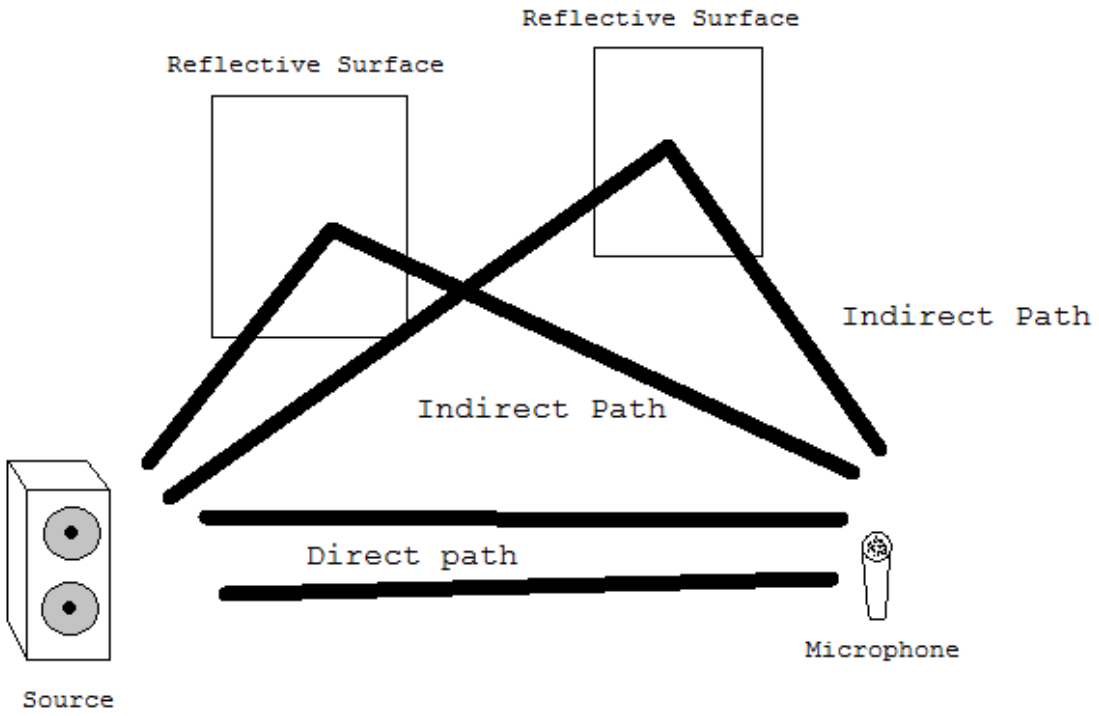

Figure 2.5: Schematic of scene affected by multipath

in the time domain, where the time index of each impulse is the time delay caused by an obstacle. Phase shifts caused by diffraction and refraction can be modeled by adding complex phase to an impulse. Given a real signal $f(t)$, the output through a channel with multipath is given by Eq. 2.2.1.1.

$$y(t) = \sum_{t=-\infty}^{\infty} \sum_{m=1}^{N} f(t) a_m e^{j\theta_m} \delta(\tau_m - t) \tag{2.2.1.1}$$

The received signal $y(t)$ is given by the convolution of the impulse response (Figure 2.6) of the multipath channel and the sounds created by the source. As the variable that represents attenuation, a_m scales a received multipath signal based on the attenuation caused by the medium and power loss that is encountered from reflections and diffrac-

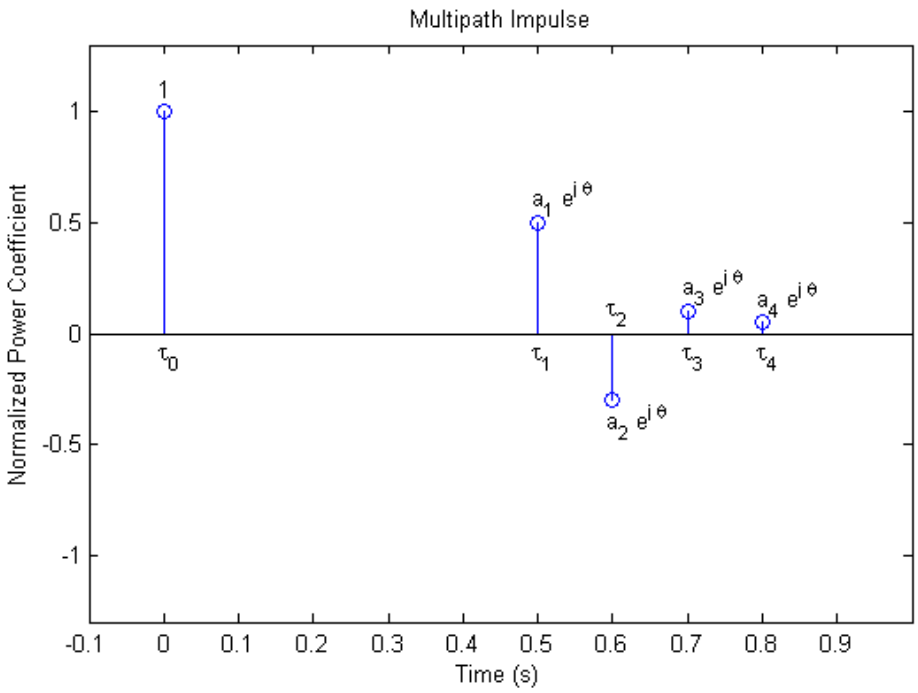

Figure 2.6: Impulse response of a multipath model

tions. θ_m represents the phase shift of a multipath signal. Lastly, τ_m is the time delay of a multipath signal. The distance and organization of an environment define the number of multipath signals and their respective time delays.

Depending on the number of paths, their time and phase delays, and sampling rate of a microphone, multipath can create intermodulation distortion. Although not significant, multipath's effects in the frequency domain are not negligible. As can be seen in Figure 2.7, the frequency response of spoken sounds appear to be smeared, as is indicated by the arrows in Figure 2.7. Since speaker recognition systems base their classifications on statistical distributions of Mel cepstral coefficients, degradation can be expected when multipath is encountered. Scenarios can arise where voiced sounds overlap, creating Mel cepstral coefficient distributions that a speaker recognition system could not have encountered during controlled or otherwise high quality enrollment sessions.

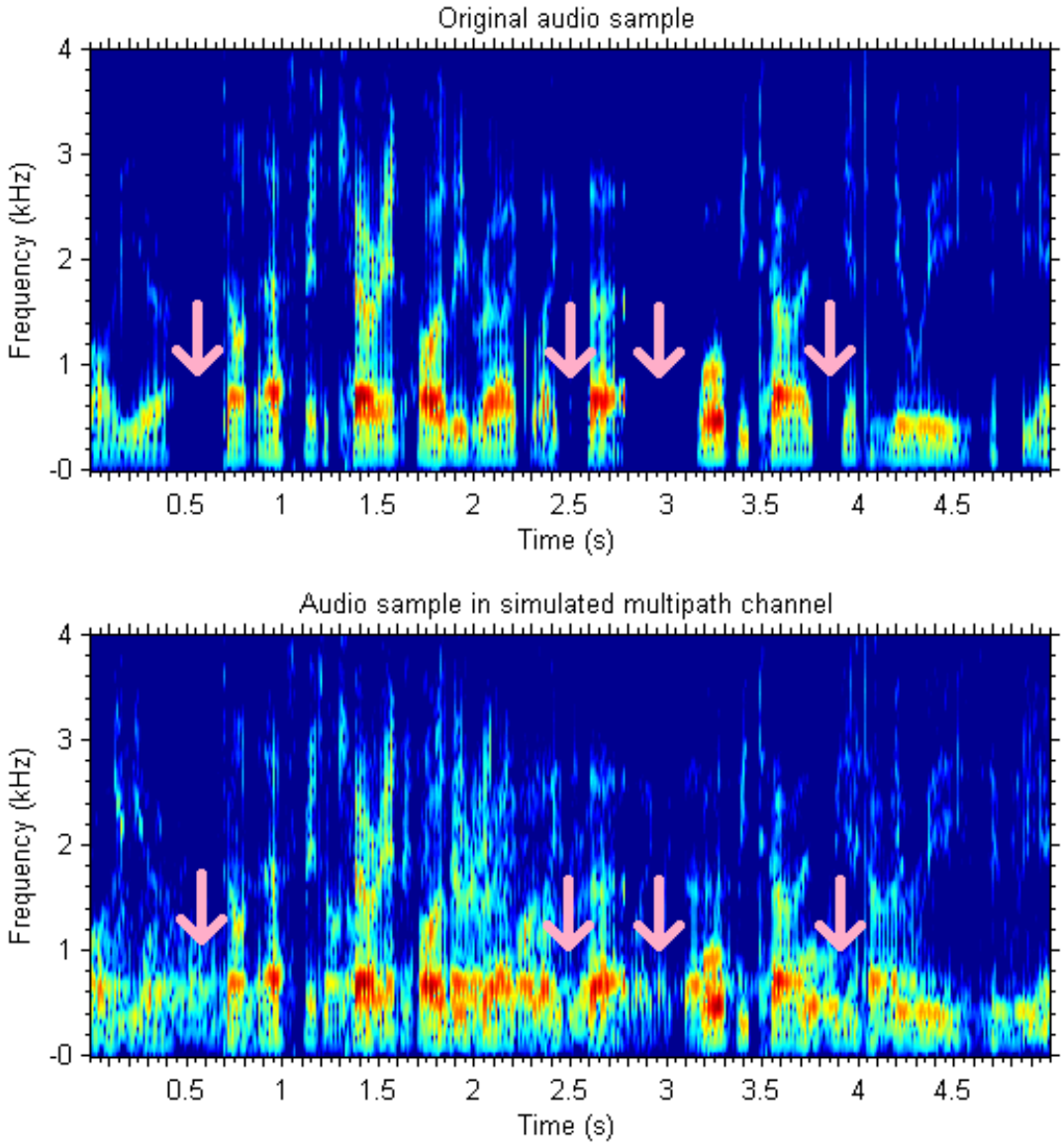

Figure 2.7: Comparison of an audio sample before and after transmission through a multipath channel

2.2.2 Speech Encoders

Speech encoders are used in bandwidth-limited applications such as cellular phone and VoIP networks. Speech encoders strive to compress audio streams into digital

bit streams based on models of human speech.

Speech prediction algorithms can be based on the source-filter model of speech production. Although the model is an approximation it is widely used due to its simplicity and computational ease [1]. The model consists of two submodels, one of the vocal cords and the other of the vocal tract. The vocal cords are assumed to be the source of a spectrally flat tone that is regarded as the excitation signal. The vocal tract acts as a filter for the excitation signal by shaping the spectrum of the tones into sounds of speech. It is the vocal cord model that speech encoders use to identify and compress voiced sounds. It is the repetitive characteristic that makes voice sounds, such as vowels, easy to identify and compress with relatively low loss. Alternatively, fricatives lack periodic excitation signals and resemble Gaussian noise.

Clear audio streams of human speech can be compressed very well with minimal loss in quality due to the linear predictive nature of the codecs. Furthermore, sounds that fall outside the bounds of human speech cannot be encoded in a way that maintains high quality. Although linear predictive encoders work well with human speech, the addition of noise to audio streams forces the encoder to encode noise it cannot model. Hence the quality of audio streams that are produced by decoding audio streams that encounter noise are highly unpredictable.

2.2.2.1 Code Excited Linear Prediction

Linear predictive coding is the core technique used by many speech encoders. Linear prediction is a systematic way of predicting a signal $x[n]$ by linear combinations of its past samples, given as $x[n-i]$, where a_i are the linear prediction coefficients.

$$y[n] = \sum_{i=1}^{N} a_i x[n-i] \tag{2.2.2.1}$$

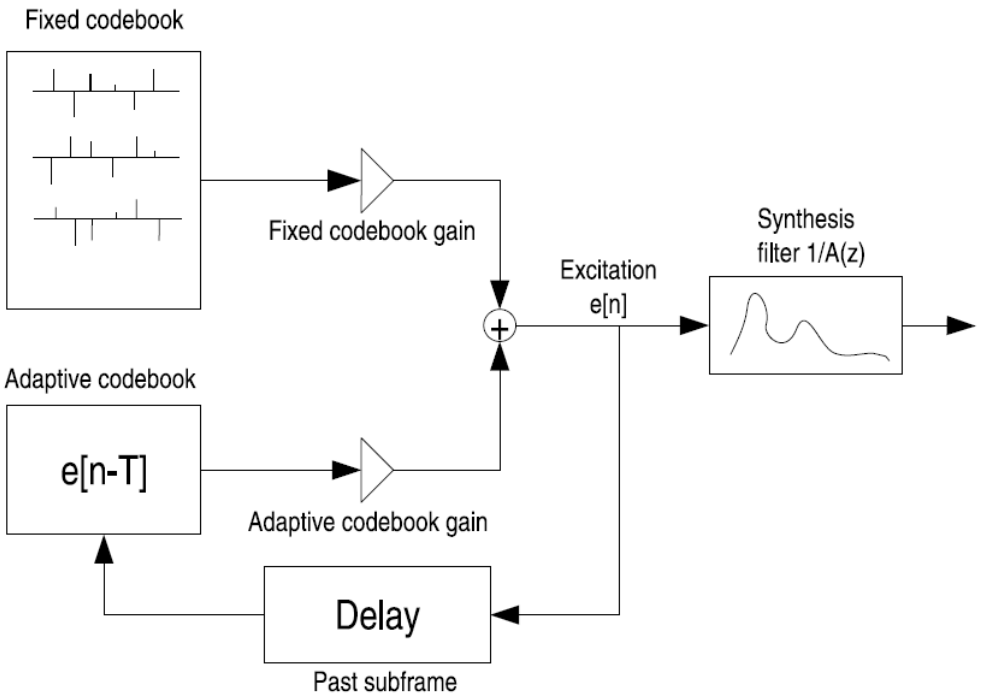

Figure 2.8: The CELP model of speech synthesis [1]

It is unlikely that a signal can be predicted without error using linear methods; therefore the error of the signal $x[n]$ and the linearly predicted signal $y[n]$ is given by Eq. 2.2.2.2, which is also known as the prediction error.

$$e[n] = x[n] - y[n] = x[n] - \sum_{i=1}^{N} a_i x[n-i] \tag{2.2.2.2}$$

LPC speech encoders attempt to minimize the quadratic error function 2.2.2.3. The quadratic error function allows for the error metric to converge to local minima solutions more

quickly when LPC calculates new prediction coefficients.

$$E[n] = \sum_{n=0}^{L-1}[e[n]]^2 = [n] - y[n] = \sum_{n=0}^{L-1}\left[x[n] - \sum_{i=1}^{N} a_i x[n-i]\right]^2 \tag{2.2.2.3}$$

Linear predictive methods are very well suited for predicting voiced sounds such as vowels due to their periodic characteristics. Since LPC can very easily compress vowels, the benefit to speaker recognition is substantial. LPC is capable of transmitting the parts of speech that allow statistical models to identify the speaker uniquely without a significant loss in quality. Since speaker recognition systems process entire audio streams and not just vowels, the expected loss in decoding quality of fricatives must be examined to establish the effects of voice encoders on speaker recognition systems.

Modern audio encoders rely on psychoacoustic findings to optimize compression ratios by using perceptually weighted mean square error functions. CELP encoders, specifically, rely on a technique called Analysis-by-Synthesis (AbS) to generate the encoded bitstreams. The error between an encoded-then-decoded audio stream is compared with the original waveform using the perceptually weighted mean square error function. Since Mel cepstral coefficients are extracted using filters that rely on the human perception of sound speaker recognition systems are heavily affected by the accuracy of these algorithms.

2.2.2.2 Adaptive Multi-Rate (AMR) Codec

The adaptive multi-rate codec is a patented speech codec that is maintained by 3GPP [23]. AMR is widely used in GSM and UMTS cellular networks around the world.

AMR is capable of changing its bitrate based on the complexity of the sounds being compressed and on the quality of the RF link. Half-rate channel bandwidths can operate as low as 4.75 Kbps while full-rate channels can operate at 12.20 Kpbs. Audio is filtered

by a bandpass 200 - 3400 Hz filter then sampled using 13 bits at 8 KHz. Samples are then segmented into frames, which consist of 160 samples. At 8 KHz that makes them 20 ms long. ACELP is used as one of the core compression algorithms. AMR features a few of noise cancelling such as Voice Activity Detection and Comfort Noise Generation.

2.2.2.3 Enhanced Variable Rate Codec (EVRC)

The Enhanced Variable Rate Codec is a widely used speech codec patented by Qualcomm that is the de facto standard in CDMA cellular networks. EVRC uses RCELP as its core compression algorithm.

Unlike other voice codecs EVRC features only three data rates: full rate, half rate, and eight rate, which operate at 8.55 Kbps, 4.0 Kbps, and 0.8 Kbps respectively [24]. EVRC samples 20 ms long audio frames using 16 bits at 8 KHz.

2.2.2.4 Speex Codec

Speex is an open source and patent free audio codec widely used in VoIP applications. Speex uses CELP algorithms to achieve data compression. Although it is not intended for cellular telephony, Speex's configuration in VoIP deployments very closely resembles the AMR and EVRC codecs. The audio channel can be configured to be sampled at 8 KHz [1]. Speex does not have any preconfigured operating bitrate settings and as such is able to operate anywhere between 4.8 Kbps and 16 Kbps.

Chapter 3

Implementation

A Gaussian mixture model based speaker recognition system was developed using the readily available open-source VoiceBox toolbox [25]. The existing functionality of the toolbox allowed for an implementation that follows standard practices for signal processing in speaker recognition systems. As such, Mel frequency feature extraction was achieved using properly tuned custom filter banks in conjunction with the cepstral functionality offered by Voicebox.

The front-end signal processing block was designed to match the fundamental sampling frequencies of training and testing data. During preliminary testing, variations in Mel frequency cepstral coefficients distributions averaging 1.21% were observed between audio samples and their decimated versions. The decision to decimate all audio samples before processing was made in order to remove the effects of decimation on the experiments that followed. Decimating audio samples to 8 KHz was also necessary to use the speech encoders. The constraint is not obligatory for a speaker recognition system; its use does not complicate the implementation of speaker recognition systems.

Further along the block diagram (Figure 1.1), the statistical based pattern matching block uses the extracted low-level MFCC features for training and testing. The statistical

classifier whose models are trained and whose output is a likelihood ratio is a Gaussian mixture model with a variable number of inner Gaussian distributions.

3.1 K-means initialized GMM

Speaker recognition systems rely on expectation maximization algorithms as a means to train a GMM by converging its parameters onto local maxima solutions. The ultimate goal of the expectation maximization algorithm is to find a set of parameters (means and variances) that minimize the difference between a GMM's curve and the actual MFCC distribution for each underlying cepstral coefficient. The algorithm requires a starting point, or a set of initial means and variances, that it then uses to converge on a local maxima; for such purposes modern implementations use random MFCC feature vectors. The reason is that if each feature vector belongs to the speaker then any convergence of expectation maximization algorithm is an acceptable solution. Alternatively, the expectation-maximization algorithm's starting conditions can be computed by determining the k-means clusters of the MFCC feature vectors. As will be shown, doing so increases reproducibility and decreases the number of iterations of the expectation maximization algorithm step. The benefit of selecting a method that provides more stable trained GMMs is important to the testing that will be described in the following chapters.

In preliminary tests, trainings that consisted of initializing the expectation maximization algorithm with means derived by performing k-means were found to converge faster and much more consistently for speakers regardless of the training data. In contrast, trainings that utilized random feature vector initialization tended to converge slower and on different local maxima from run to run. By using k-means as the starting condition for the expectation maximization algorithm, the speaker recognition system was found to converge in a third of the number of expected maximization steps needed by a random vector

initialized training. Additionally, k-means were found to execute in a quarter of the processor execution time of a random vector initialized test. This indicates that despite the extra overhead of the k-means algorithm, the system overall performed faster.

The aforementioned varying local maxima were first encountered during development of the speaker recognition system. The shape and hence the means of the underlying Gaussians within the GMM varied widely between training sessions with identical but randomly shuffled audio datasets. Two distinct randomly initialized GMMs are show in Figure 3.1 and Figure 3.2. Additionally, two distinct k-means instantiated GMMs are shown in Figure 3.3 and Figure 3.4. In each of the four graphs, the embolded lines represent the curve that is generated by summing each GMM's underlying Gaussian distributions according to their respective weights. The thinner, low laying curves are the outlines of each underlying Gaussian distribution. Ultimately, the histograms in the background of each subgraph show the distribution of MFCCs on a per cepstral coefficient basis.

Ideally, multiple audio samples of a speaker would produce identical GMMs. This however is not case, a quick per-MFCC comparison of the shape of GMMs, the bold curves in each subgraph, in Figure 3.1 and Figure 3.2 shows variations even though the speaker and the audio samples remain unchanged. The subgraphs for MFCCs 4 and 8 show stark differences. Subsequently, an identical comparison of the k-means initialized GMMs in Figure 3.3 and Figure 3.4 shows far less variance.

A statistical investigation into the improvements brought about by the use of k-means initialization is presented. The investigation specifically targeted the improvements to middle MFCCs whose bandwidths are the most important for human understanding of speech according to psychoacoustics. The improvement to each MFCC was determined by observing the means and variances of the GMMs's underlying Gaussian distributions. The data for the analysis was generated by averaging the means and variances collected from 1,000 trials of training GMMs with audio samples that consisted of randomly shuf-

fled sub audio samples.

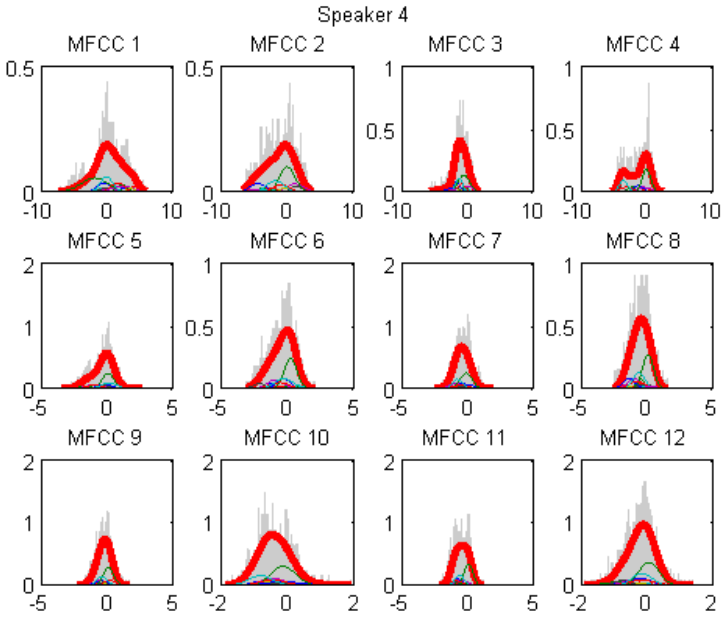

Figure 3.1: Random vector intiliazed GMM

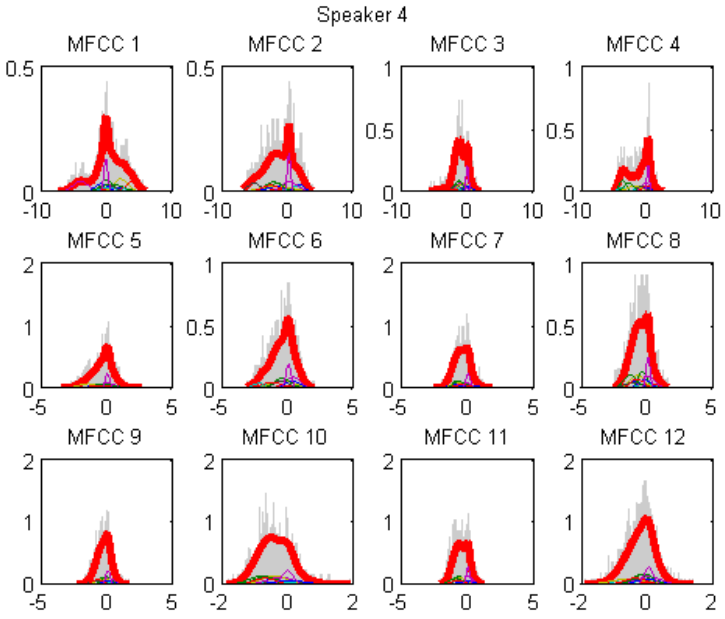

Figure 3.2: Random vector intiliazed GMM

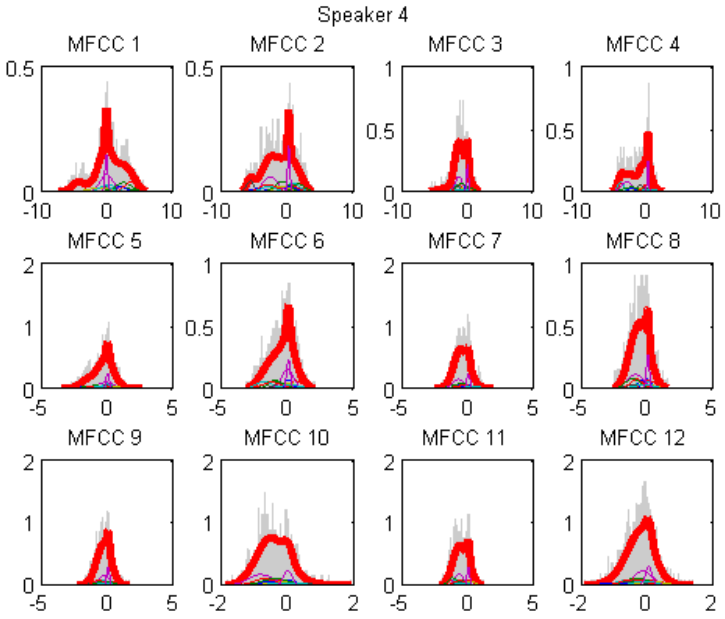

Figure 3.3: K-means initialized GMM

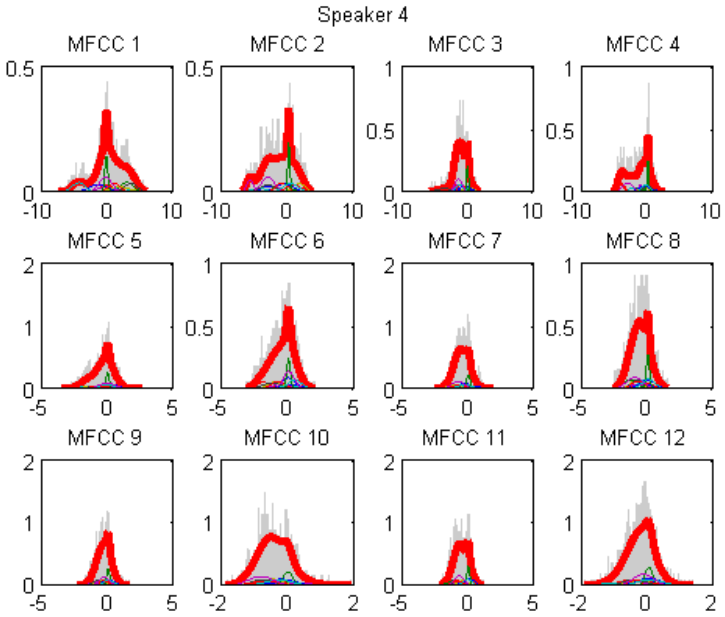

Figure 3.4: K-means initialized GMM

Means	Variance
-2.7418	0.3850
-1.9511	0.6499
-1.5745	0.5780
-1.1540	0.2426
-0.8906	0.1475
-0.7834	0.1142
-0.6578	0.0956
-0.4536	0.2108
-0.3233	0.1377
-0.1637	0.1588
-0.0163	0.1192
0.2720	0.2869

Table 3.1: Means and variances of the means of the 3rd MFCC's underlying Gaussian distributions within a GMM

Means	Variance
-3.1014	0.2883
-2.3314	0.1885
-1.4432	0.4358
-1.0651	0.1359
-0.9367	0.0627
-0.7945	0.0786
-0.6829	0.0622
-0.6261	0.0228
-0.2676	0.0996
-0.0780	0.1345
0.0179	0.0128
0.1157	0.1005

Table 3.2: Means and variances of the means of the 3rd MFCC's underlying Gaussian distributions within a GMM initialized by k-means

The third and fourth MFCCs contain the greatest amount of information due to frequency bandwidth extending from 200 to 360 Hz. It is essential that these MFCCs experience the least amount of variance due to the importance of their stability. The average variance of the means of the third MFCC GMM's underlying Gaussian distributions in a random vector intialized GMM (Table 3.1) is 0.265. Comparatively, the variance for the same MFCC and identical speaker with the same audio sample in a k-means intialized GMM (Table 3.2) is 0.1352.

Means	Variance
-2.9300	0.3577
-2.3795	0.1811
-1.7172	0.4174
-1.3748	0.3054
-1.0181	0.2591
-0.8597	0.2225
-0.6146	0.1500
-0.4352	0.1490
-0.3040	0.1021
-0.1967	0.1324
-0.0463	0.1149
0.1669	0.3237

Table 3.3: Means and variances of the means of the 4th MFCC's underlying Gaussian distributions within a GMM

Means	Variance
-3.2148	0.0177
-2.7209	0.1514
-1.8416	0.3618
-1.3056	0.3347
-1.0495	0.1181
-0.9295	0.1165
-0.7496	0.1420
-0.5234	0.0938
-0.4089	0.0847
-0.2248	0.1452
-0.0051	0.0661
0.0414	0.0111

Table 3.4: Means and variances of the means of the 4th MFCC's underlying Gaussian distributions within a GMM initialized by k-means

The average variance of the means of the 4th MFCC GMM's underlying Gaussian distributions in a random vector intialized GMM (Table 3.3) is 0.2263. Comparatively the variance for the same MFCC and identical speaker with the same audio sample in a k-means intialized GMM (Table 3.4) is 0.1369.

Means	Variance
-1.6760	0.3486
-1.0175	0.2697
-0.7604	0.1776
-0.5987	0.0851
-0.4887	0.1303
-0.3682	0.1695
-0.2793	0.1871
-0.1647	0.1899
-0.0348	0.1691
0.0714	0.0960
0.3321	0.1884
0.7020	0.3189

Table 3.5: Means and variances of the means of the 5th MFCC's underlying Gaussian distributions within a GMM

Means	Variance
-1.8456	0.2020
-1.0003	0.1792
-0.7957	0.1232
-0.6558	0.0442
-0.5702	0.0411
-0.4873	0.0985
-0.3401	0.0721
-0.2432	0.1173
-0.0864	0.0796
0.0403	0.0544
0.4412	0.2532
0.6476	0.0803

Table 3.6: Means and variances of the means of the 5th MFCC's underlying Gaussian distributions within a GMM initialized by k-means

The average variance of the means of the 5th MFCC GMM's underlying Gaussian distributions in a random vector intialized GMM (Table 3.5) is 0.1942. Comparatively the variance for the same MFCC and identical speaker with the same audio sample in a k-means intialized GMM (Table 3.6) is 0.1121.

Means	Variance
-2.2939	0.3121
-1.3708	0.4824
-0.9681	0.1518
-0.8722	0.1005
-0.6826	0.2018
-0.4747	0.2588
-0.1419	0.2176
0.0294	0.0541
0.2209	0.1848
0.4108	0.1825
0.6265	0.1712
0.8770	0.1489

Table 3.7: Means and variances of the means of the 6th MFCC's underlying Gaussian distributions within a GMM

Means	Variance
-2.2741	0.2622
-1.1541	0.2219
-0.9631	0.0806
-0.8007	0.1135
-0.6211	0.2153
-0.2761	0.2761
-0.1154	0.1946
0.0086	0.0446
0.1233	0.1089
0.3564	0.0907
0.6193	0.2053
0.8571	0.0740

Table 3.8: Means and variances of the means of the 6th MFCC's underlying Gaussian distributions within a GMM initialized by k-means

The average variance of the means of the 6th MFCC GMM's underlying Gaussian distributions in a random vector intialized GMM (Table 3.7) is 0.2055. Comparatively the variance for the same MFCC and identical speaker with the same audio sample in a k-means intialized GMM (Table 3.8) is 0.1573.

3.1.1 Remarks

MFCC	Randomized GMM	K-means GMM
1	0.1918	0.1349
2	0.1865	0.1622
3	0.1992	0.1660
4	0.2494	0.1820
5	0.2231	0.1448
6	0.2574	0.1483
7	0.1987	0.1511
8	0.2110	0.1200
9	0.2588	0.1548
10	0.2087	0.1490
11	0.2364	0.1159
12	0.1921	0.1314

Table 3.9: Variances of the means of MFCCs's underlying Gaussian distributions within GMMs in a group of speakers

The improvement was tested against a group of speakers. The decrease in trained GMM variations is shown in Table 3.9, which shows the averages of the observed variances found in the underlying means of the Gaussian distributions within the GMMs on a per MFCC basis for the whole group. Variances in the MFCCs whose ranges encompass the key bandwidth of human speech (MFCCs 3 through 6) are the locations where the k-means instantiated GMMs outperform the randomly instantiated GMMs on average by nearly a factor of two. The lower variances indicate a higher stability in the training values and thus the results of the system. After an analysis of the improvements in results, the k-means initialized GMM implementation was selected in place of random vector initialization for the purposes of this thesis.

Chapter 4

Proposed Metric

The only metric currently available for testing speaker recognition systems is the NIST-SRE metric [12] which is a simple tabulation of error rates of a speaker recognition system under varying training and testing audio length, and other conditions.

The first output of a speaker recognition system comes from the statistical classifier, which in this case is a GMM. Afterwards a maximum function finds the speaker model with the highest log-likelihood score and ultimately presents the speaker of that model as the identified speaker. In the process of the classifier selecting the model with the highest log-likelihood score as the identified speaker, intermediary information such as the log-likelihood difference between speaker models with the two greatest scores is lost. The difference is key to understanding the quality of identifications made by a speaker recognition system.

Log-likelihood scores are greatly influenced by peaks. Peaks in the log-likelihood output of a trained GMM occur periodically throughout an audio sample because the MFCC distribution of a speech vector only seldom correlates perfectly with the GMM's probability distribution function. Peaks are essential for correctly identifying speakers since peaks contribute orders of magnitude more to the total log-likelihood score upon which

matches are based.

Note must be taken that linear differences in log-likelihood values correspond to exponential changes. Therefore it is essential to know whether error rates vary due to changes in the occurrence and impact of peaks or if they are caused by changes to the log-likelihood value floor that corresponds to non-matches.

4.1 Applicability

A situation may arise where a proposed improvement to a speaker recognition system might improve the overall error rate and simultaneously decrease the confidence of the identification [26]. The confidence, or quality, of an identification is the difference between the two greatest log-likelihoods. Such a situation may go undetected due to improved error rates but as the number of speaker candidates increases those margins will keep the error rates at predictable levels. Additionally, the proposed metric will be used as a metric function for a genetic algorithm that mutates weights representing the importance of MFCCs. By altering the weights of MFCCs, speaker recognition systems facing channel impediments can be improved by decreasing their reliance on MFCCs affected by noise. The metric's purpose is to help the genetic algorithm detect improvements while avoiding local-minima and shortsighted mutations that might ultimately lead to poor performance. It is assumed that the metric will perform better than the NIST-SRE metric because of its ability to quantifiably determine the quality of identifications.

4.2 Theory

The genetic algorithm that has been employed is a variant of the Tabu mutation technique. At the start of each generation all of the existing variables can be changed however

changes to variables that yield worse results are added to a list denoting "taboo" genes, which causes the respective variable to not be eligible to be altered until the start of the next generation. With the use of a proper metric, Tabu based genetic algorithms are much more likely to avoid taking sub-optimal local maxima paths [27]. It is the metric's ability to gauge the quality of the statistical matches based on low level statistical data that theoretically allows it to outperform the NIST-SRE. Since the metric is designed for use with Tabu based genetic algorithms, it is more than qualified to work with unenhanced genetic algorithms as well.

The proposed metric, shown in Eq. 4.2.0.1, is calculated from the mean square error of the moving average of the log-likelihood output of a trained GMM between a control group log-likelihood ($channel_{control}$), which encounters no channel impediments, and a test group log-likelihood ($channel_{noisy}$), which is the channel exposed to channel impediments. The metric creates a condition in which alterations to log-likelihood curve peaks greatly impact the metric score while changes in the signal noise floor only affect the metric score linearly if the change is small enough. Peaks that are degraded are as damaging to the system as erroneously created peaks; thus the system does not differentiate.

$$M.S.E. = \sum_{t=W+1}^{T} \frac{1}{W} \sum_{w=1}^{W} \|channel_{control}(t-w) - channel_{noisy}(t-w)\|^2 \qquad (4.2.0.1)$$

The decision to filter the log-likelihood curves with a moving average was made to make the metric more closely resemble human perception by having it ignore short-term variations in speech. As such, the length of the moving average window was experimentally determined. The audio samples used in the experiment were created by having speakers vocalize a single vowel phoneme for 5 seconds. Once selected the 5 second long audio samples were bisected; the two halves of the audio sample were then utilized as

both $channel_{control}$ and $channel_{noisy}$. Since the audio samples were subjectively considered by human listeners as being identical the moving average window size was increased until the metric no longer indicated a noticeable difference between the audio samples. It is important for a speaker recognition system to detect as little noise as possible from the log-likelihood of a trained GMM when a speaker purposely pronounces a phoneme unchanged because it removes the log-likelihood noise the system will encounter as the person is speaking naturally.

The metric (Eq. 4.2.0.1) is intended to be used as a means of quantifying the effect of alterations to a speaker recognition system during development and testing. By simulating channel impediments, an audio sample passed through a noisy channel can have its log-likelihood curve compared against the clear audio sample's log-likelihood by using the metric. The resultant value is a relative quantified value of the effect on the speaker recognitions system. Furthermore, the metric can be used to quantify the improvements of modifications to speaker recognition systems. This can be accomplished by computing the control log-likelihood ($channel_{control}$) of a clear audio sample by passing it through an unmodified speaker recognition system. Consequently this allows $channel_{noisy}$ to be the log-likelihood curve of audio samples affected by channel impediments passed an altered speaker recognition system. In the following chapter, these alterations will be varying MFCC weights. Therefore, it is possible to compare two modified speaker recognition systems by comparing the two values the proposed metric computes when each of the two variations to the speaker recognition systems have their log-likelihood outputs substituted for $channel_{noisy}$ and compared against the clear audio sample's log-likelihood, $channel_{control}$. The modified speaker recognition system with the score closest to 0 is considered to be better because its modifications bring the system closer to performing as well as when there is no noise.

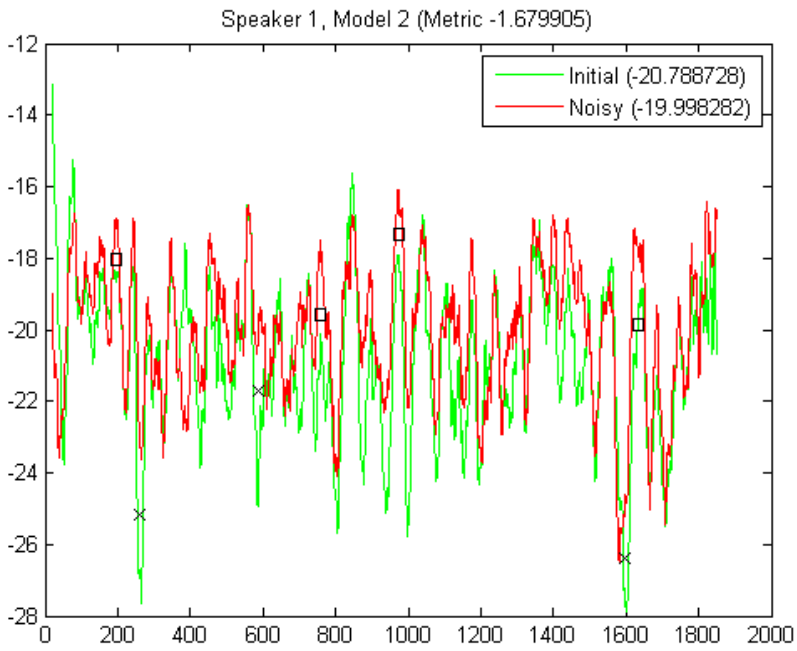

Figure 4.1: Log-likelihood output of a speech samples and models from different speakers

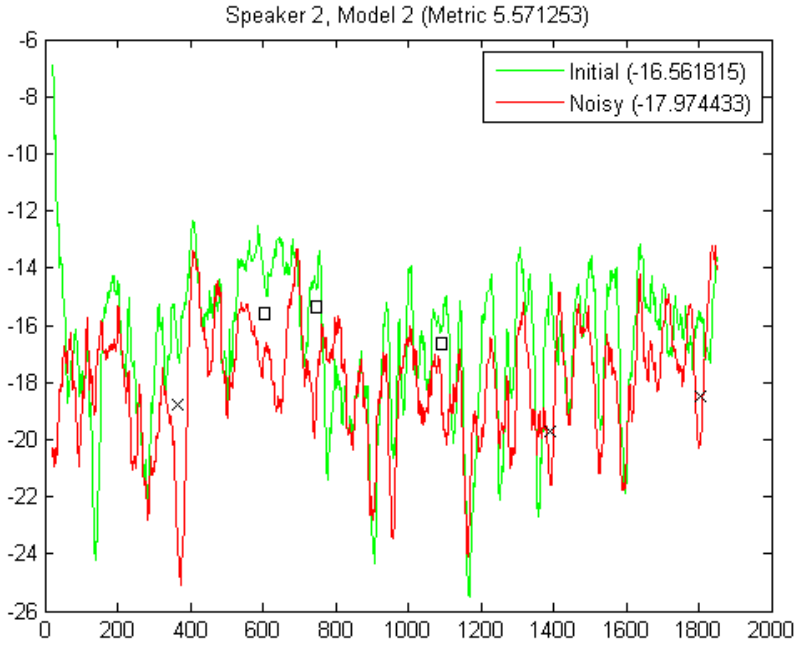

Figure 4.2: Log-likelihood output of a speech samples and models from identical speakers

The effects of 6dB SNR of white noise on the outputs of a trained GMM are shown in Figure 4.1 and Figure 4.2. Two speaker models were trained with clear speech samples from the TIMIT corpora. Figure 4.1 shows the log-likelihood of two audio samples (one noisy and one clear) belonging to speaker 1 being evaluated against speaker 2's trained model. Conversely, Figure 4.2 shows the log-likelihood of two audio samples (one noisy and one clear) belonging to speaker 2 being evaluated against speaker 2's trained model. Green lines are the log-likelihoods computed from unaltered speech samples, and red lines are the log-likelihoods computed from the sample audio samples with 6 dB of white noise.

Figure 4.2 shows a sample log-likelihood output from a test scenario where a GMM was trained and tested using audio samples from the same speaker. The log-likelihood extracted from a clear audio sample (green line) has a much higher mean (-16.56) than the log-likelihood extracted from the same audio sample at 6dB SNR, which has a mean of -17.97. Since the means are calculated from logarithmic scores and the plots are log-scale representations of the likelihood outputs of the GMMs, these differences are significant. In Figure 4.2 some peaks that could help the speaker recognition system identify speaker 2 correctly were lost in several instances (marked by squares). Additionally, the crosses indicate feature vector sets that were significantly damaged by the noise, in these cases the peaks that could have contributed to identifying the speaker turned into negative peaks that very negatively influence the log-likelihood score. Figure 4.1 shows a sample log-likelihood output from a test scenario where a GMM was trained and tested using audio samples from the different speakers. The log-likelihood extracted from a clear audio sample (green line) has a much lower mean (-20.78) than log-likelihood extracted from the same audio sample at 6 dB SNR, which has a mean of -19.99. In this case the mean being closer to zero is less desirable because that would cause speaker 2's trained GMM to falsely identify speaker 1. In Figure 4.1 the squares indicate peaks that were mis-

takenly identified as better than average matches that were then increased by the noise. Additionally, the crosses indicate peaks that correspond to feature vector sets that were significantly damaged by the noise and are no longer contribute positively to the final score. Comparing Figure 4.1 and Figure 4.2, it is clear that noise behaves in a destructive manner. In Figure 4.1, the red line generally is above the green line, causing speaker 1's noisy audio sample to be a closer match to speaker 2's model. Conversely in Figure 4.2, the red line is below the green line, causing speaker 2's noisy audio sample to be a worse match to their own model than the clear audio test.

4.3 Preliminary testing

The effectiveness of the proposed metric over the NIST-SRE metric was established by having each error metric act as the Tabu based genetic algorithm 's error function. The performance of the genetic algorithm was quantified by its ability to find MFCC weight arrangements that benefitted the speaker recognition system's error rate; the recorded data included the number of attempted, and successful mutations. The MFCC weight vectors generated by the genetic algorithm can be conceptually interpreted as a reliability index of an MFCC given a certain channel impediment. In the simple case of flicker noise, where most of the energy of the noise is in the lower frequency range, the genetic algorithm gives the first 4 MFCCs less weight when generating the log-likelihood for speech samples.

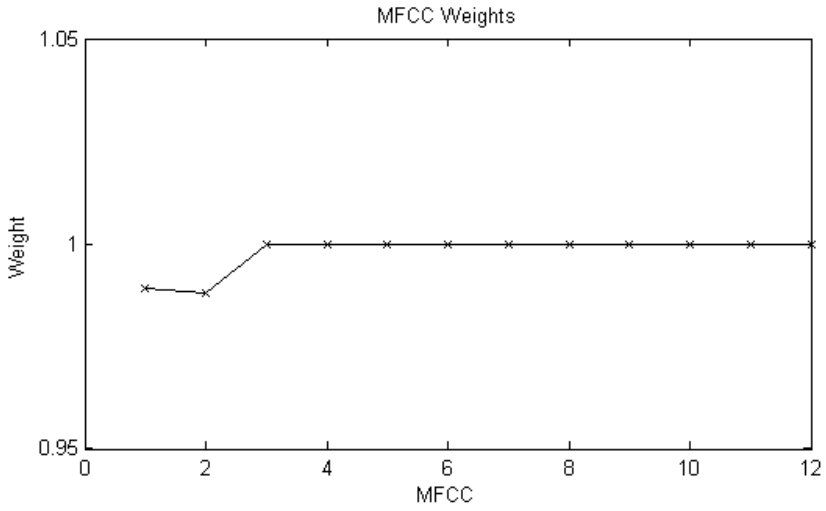

Figure 4.3: MFCC weights of a system exposed to noise centered at 150Hz

Testing was conducted by band-pass filtering white noise with a bandwidth of 250Hz at various center frequencies. The target SNR for each test was about 10dB, this is where a 50% error rate was observed. The expected output of each test was to observe MFCCs whose bandwidths overlapped with the band-pass filtered noise to decrease to below unity, and for other MFCCs weights to remain at unity of slightly increase. An example of this is shown in Figure 4.3. The MFCC weights show the results of genetic algorithm has modifying the relative weights of the MFCCs to rely less on the lower MFCCs when noise centered at 150Hz is introduced. Additionally, the expectation was that the proposed metric would allow the genetic algorithm to converge faster while requiring fewer mutation attempts, and ultimately yielding higher error rates than the NIST-SRE before the system reached a stall condition. A stall is caused by exceeding a predefined number of consecutive unsuccessful mutations.

4.4 Remarks

$$Improvementpercentage = 100 * \frac{ImprovedRate - BaseRate}{BaseRate} \tag{4.4.0.2}$$

Center Frequency	Base error rate	Improved error rates	Improvement percentage
$150Hz$	29.7429	30.1096	1.2329%
$200Hz$	28.3714	28.4571	0.3020%
$400Hz$	29.0571	29.4324	1.2916%
$700Hz$	29.7429	29.5714	-0.5766%
$1000Hz$	30.1714	30.7714	1.9886%
$1500Hz$	30.4286	31.0286	1.9718%
$2200Hz$	30.2571	30.1714	-0.2832%

Table 4.1: Improvements by genetic algorithm using NIST SRE

Center Frequency	Base error rate	Improved error rates	Improvement percentage
$150Hz$	29.4857	33.2286	12.6939%
$200Hz$	29.0571	32.6571	12.3894%
$400Hz$	28.7143	33.6857	17.3133%
$700Hz$	29.9143	31.9714	6.8766%
$1000Hz$	29.7429	33.0857	11.239%
$1500Hz$	30.0857	33.8571	12.5355%
$2200Hz$	30.4286	34.9714	14.9294%

Table 4.2: Improvements by genetic algorithm using proposed metric

The results from $1,000$ trial runs for each frequency are tabulated and shown in Tables 4.1 and 4.2. The base error rate is the error rate of the unadultered speaker recognition system when encountering noise implanted at various frequencies. The improved error rate is the error rate of the speaker recognition system after its MFCC weights have been modified by the genetic algorithm using one of the two metrics. The improvement percentage is given by Eq 4.4.0.2.

From the tabulated results, it is apparent that the proposed metric almost always defeats the NIST SRE as the optimal genetic algorithm metric function when adjusting MFCC weights. Although the improvements are substantial, the reason for NIST-SRE's poor performance is its inability to detect changes occurring to the underlying classifier as it tries to minimize the training sets' error rates. The improvements achieved by the NIST-SRE are averages, and as such, on occasion, improvements that lower training error rates as much as the proposed metric will be found but at a much more diminished rate. The

proposed metric, however, is much more consistent in being able to detect improvements that will positively affect the testing error rates as well. Interestingly enough, the average improvements observed by the proposed metric are on the same order of magnitude as the few high-performing improvements that seldom occur in the NIST-SRE trials.

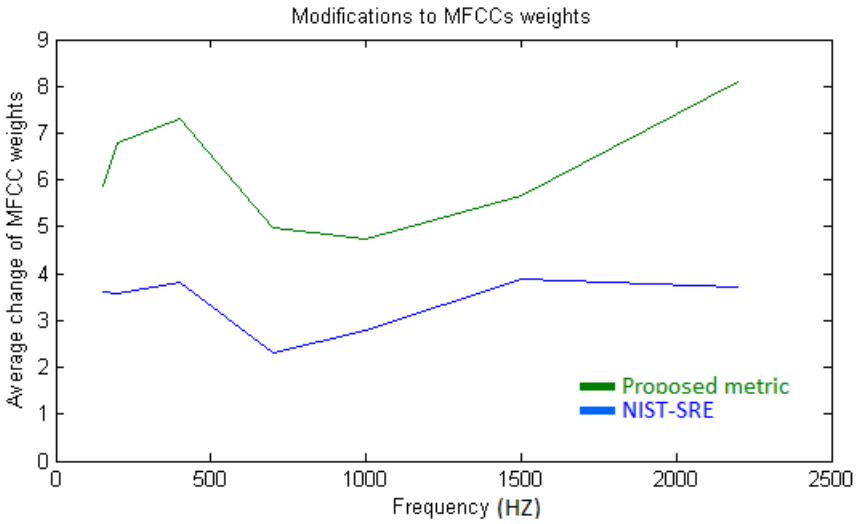

Figure 4.4: Average amount of total modifications to MFCC weights according to frequency.

Figure 4.4 shows the sum of the absolute value of all of the deviations from unity MFCC weighing. The sum can be viewed as a metric to determine the breadth of the search space the genetic algorithm was able to cover before ultimately stalling. The proposed metric is observed as being able to cover a greater breadth than the NIST-SRE based tests.

Ultimately, the behavior of the genetic algorithm is a valuable clue to the reliability and quality of its error metric. The difference between the two metrics lays in the speed at which the proposed metric is able to find a successful mutation. Comparing Table 4.3 and Table 4.4 it is apparent that the proposed metric is able to find a successful mutation in half the number of attempted mutations than the NIST-SRE metric. The number of total mutation attempts is yet another way of ascertaining the breadth of the search space a trial

Center Frequency	First successful mutation	Mutations attempts	Successful mutations
$150Hz$	88.3072	218.1714	3.2832
$200Hz$	116.2151	217.2857	3.0141
$400Hz$	102.4181	220.3143	2.8408
$700Hz$	164.0489	228.5714	2.8412
$1000Hz$	123.2542	235.0000	3.7275
$1500Hz$	132.9911	217.5143	3.0159
$2200Hz$	109.2113	234.2286	3.2780

Table 4.3: Averages of mutation results from 1,000 trials by genetic algorithm using NIST SRE

Center Frequency	First successful mutation	Mutations attempts	Successful mutations
$150Hz$	42.3737	403.1429	10.9377
$200Hz$	51.2082	361.6286	12.3886
$400Hz$	42.2659	331.9429	12.8304
$700Hz$	38.7451	336.4286	9.1875
$1000Hz$	61.0396	374.2857	10.2038
$1500Hz$	49.8458	359.8286	11.3714
$2200Hz$	52.0916	385.9429	12.0950

Table 4.4: Averages of mutation results from 1,000 trials by genetic algorithm using proposed metric

covers. Comparing the figures the proposed metric is capable of exploring much further than the NIST SRE metric. Lastly, the proposed metric is able to make more successful attempts because it considers mutations that do not immediately increase error rates as successful.

Chapter 5

Testing

Testing was conducted to further illustrate the effects of channel impediments upon modern speaker recognition systems and to exemplify the improvements brought about by the genetic algorithm using the proposed metric. Testing will explore noise models at varying SNR levels and channel models at varying compression qualities.

Channel impediments are tested by improving a speaker recognition system's performance through the Tabu based genetic algorithm when it is subjected to the impediment. As was previously discussed in Chapter 4, the genetic algorithm is configured to stop when it performs a maximum number of unsuccessful mutations, known as a stall condition. The genetic algorithm is trained using 30 speaker audio samples from the TIMIT speech corpora. Another distinct set of 30 speaker audio samples is then used for testing purposes once the genetic algorithm finishes its execution.

Each genetic algorithm execution produces three data sets, a vector of mutated MFCC weights, a score generated by the proposed metric, and the error rate of the speaker recognition system with the test audio samples before and after the execution of the genetic algorithm. For each channel impediment the data is represented in a set of four graphs that are the averages of $1,000$ runs of the genetic algorithm with identical test parameters.

The MFCC weight graph shows the weights the genetic algorithm has found to be most beneficial for increasing the speaker recognition system's performance. The error rates of the speaker recognition system with the test audio samples before and after the execution of the genetic algorithm are then presented. The error rates represent the percentage of correct identifications the speaker recognition system is able to compute. The effectiveness of the proposed metric and the genetic algorithm can then be deduced by comparing the base error rate, which occurs before the genetic algorithm mutates any of the MFCC weights, and the error rate after the genetic algorithm executes. Thirdly, the average score of the proposed metric for each speaker is recorded when the genetic algorithm stalls. The score serves to illustrate performance of the mutated speaker recognition system encountering channel impediments versus the performance of the unmutated speaker recognition system not encountering any channel impediments. The closer the score is to 0, the closer the system is to performing as well as when it does not encountering any noise, which is the purpose of mutating the MFCC weights. Ultimately a value referred to as work is presented. The value represents the total sum of each MFCC from the starting condition. Work can be interpreted as a relative amount of effort the genetic algorithm performs to achieve improvements. In case a single variable, such as SNR, is varied between trial runs, the amount by which MFCC weights change (work) is affected directly by the intensity of the channel impediment.

5.1 Noise Models

5.1.1 Testing of White Noise

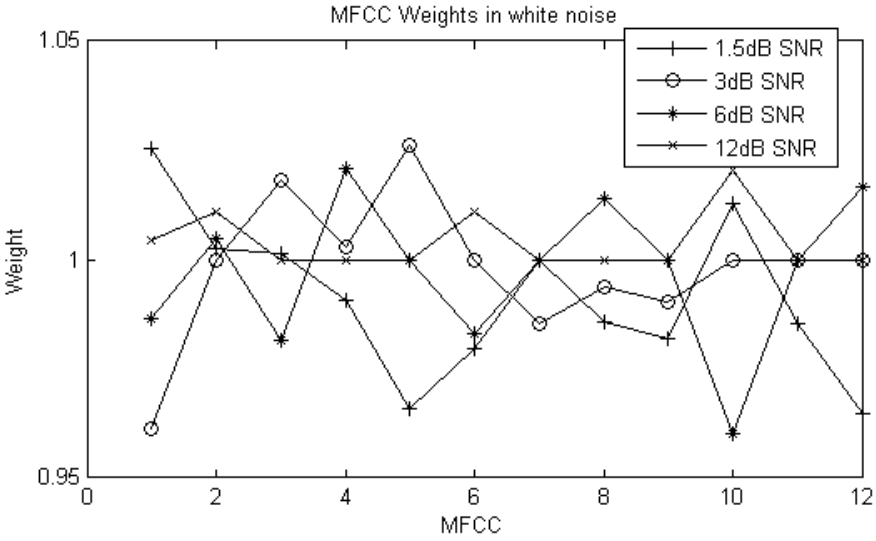

Figure 5.1: MFCC weights in white noise

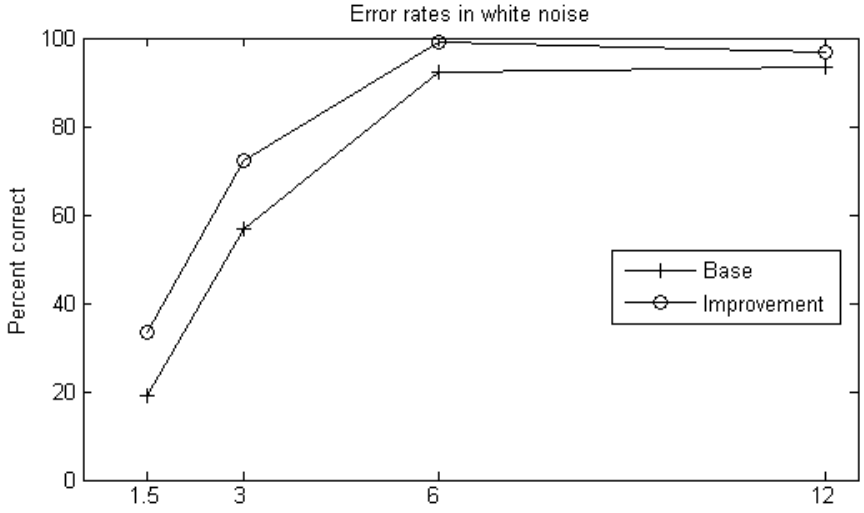

Figure 5.2: Error rates in white noise

Figure 5.1 shows the weighted MFCC output of the genetic algorithm. Although the changes in MFCC weights are sizeable their distributions appear as random as the noise

source itself. Due to the unpredictability of white noise, not much can be done to improve the error rate by altering MFCC weights. Subsequently, as the SNR increases the error rate of the system improves, as shown in Figure 5.2.

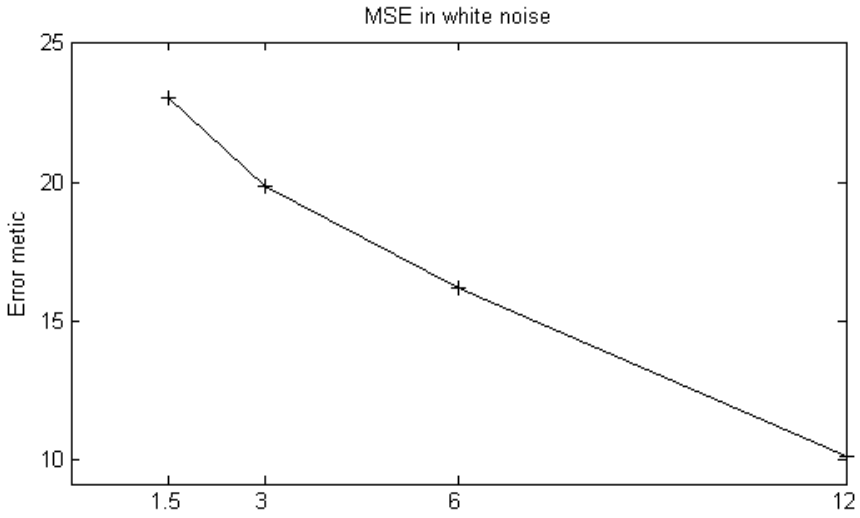

Figure 5.3: Error metric value in white noise

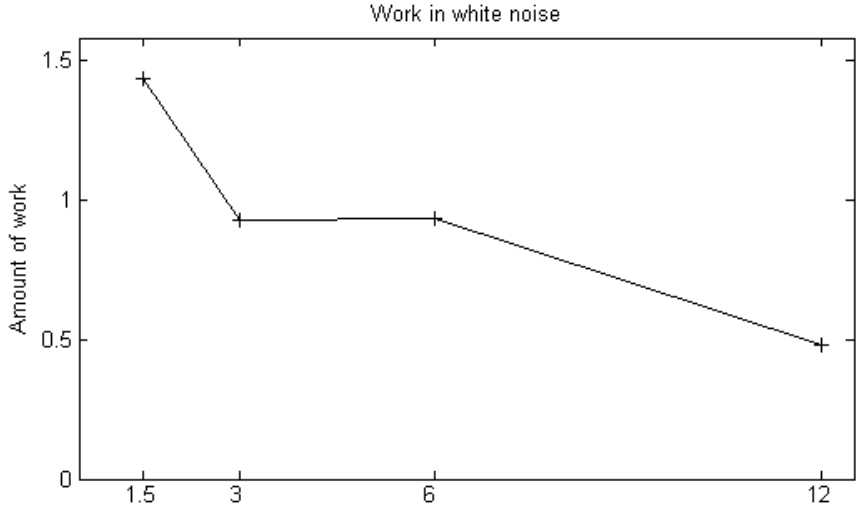

Figure 5.4: Total amount of work in white noise

Although a limited improvement in the metric is found in Figure 5.4 it is not capable of improving the system against white noise, as is expected. The power of the mutations

decreases as the level of the noise decreases.

5.1.2 Testing of AWGN

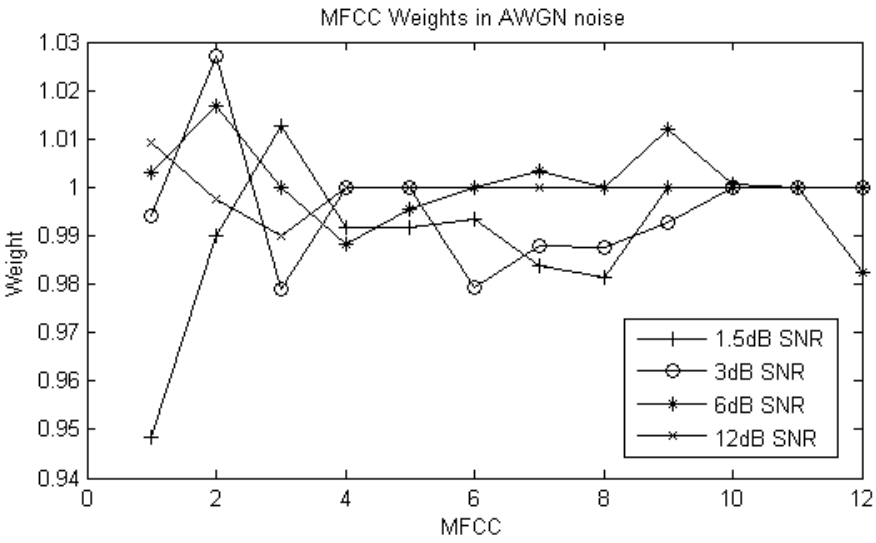

Figure 5.5: MFCC weights in AWGN noise

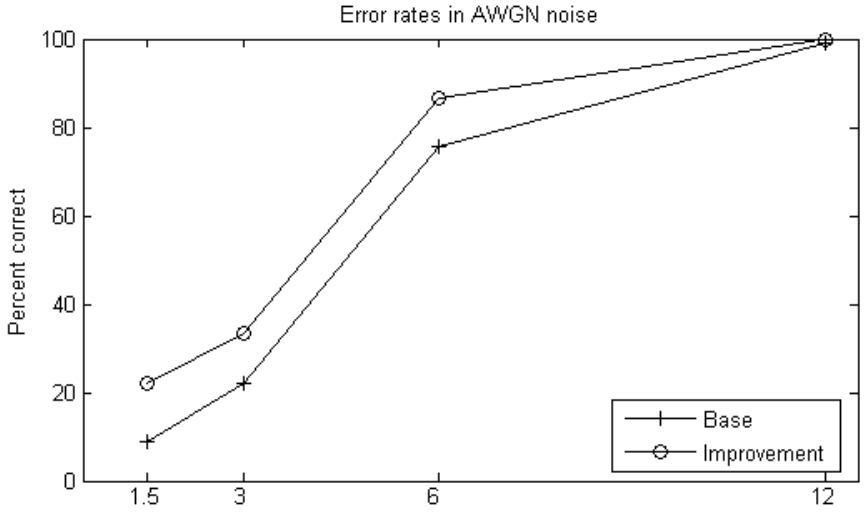

Figure 5.6: Error rates in AWGN noise

The MFCC weights in Figure 5.5 for additive white Gaussian noise do not different significantly from white noise, shown in Figure 5.1. The AWGN noise error rate also cannot be improved due to the unpredictability of the noise. Error rates were found to improve as the SNR increases, as shown in Figure 5.6.

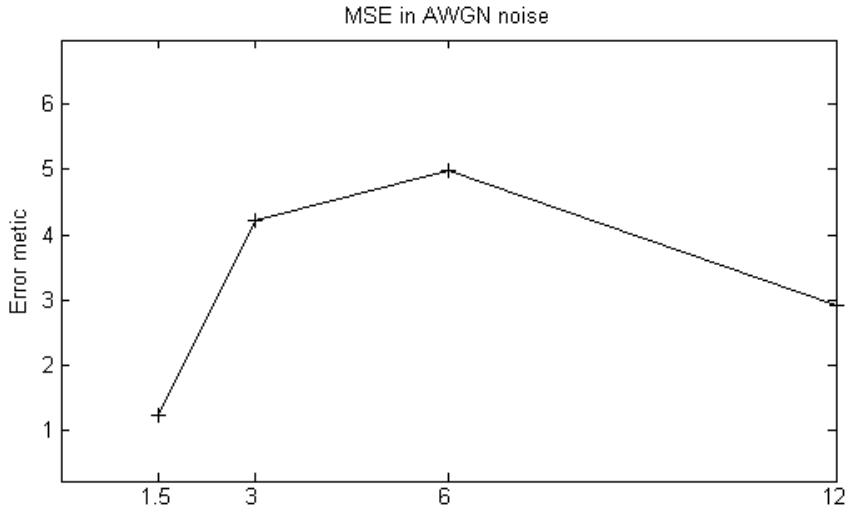

Figure 5.7: Error metric value in AWGN noise

The error metric curves of white noise, shown in Figure 5.3, and additive white Gaussian noise, shown in Figure 5.7, appear identical.

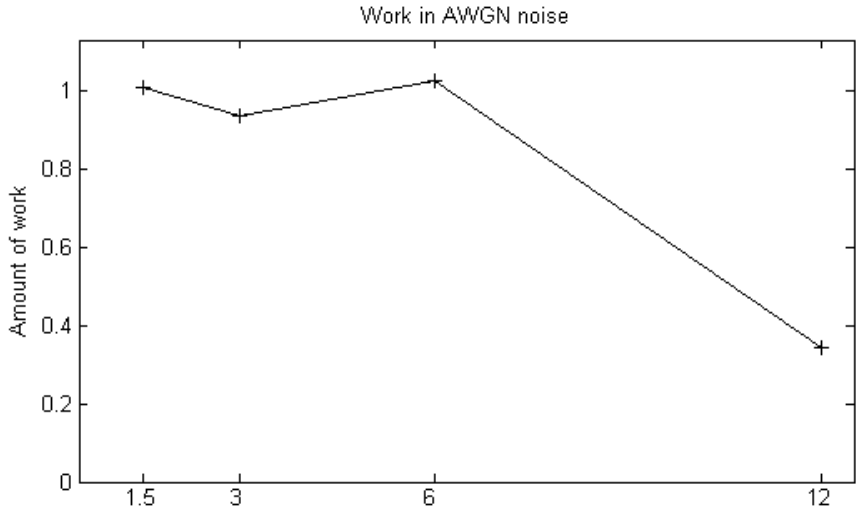

Figure 5.8: Total amount of work in AWGN noise

The work curves appear to experience the same effects as the metric curves. Due to the similarity of the white noise and additive white Gaussian noise, it can be concluded that the underlying numerical process of the white noise source does not influence the

performance of speaker recognition systems.

5.1.3 Testing of Pink Noise

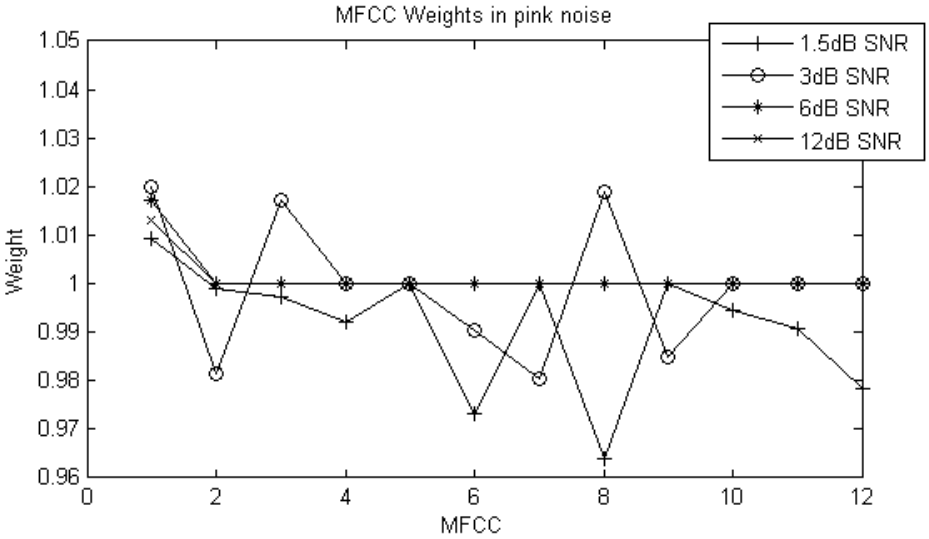

Figure 5.9: MFCC weights in pink noise

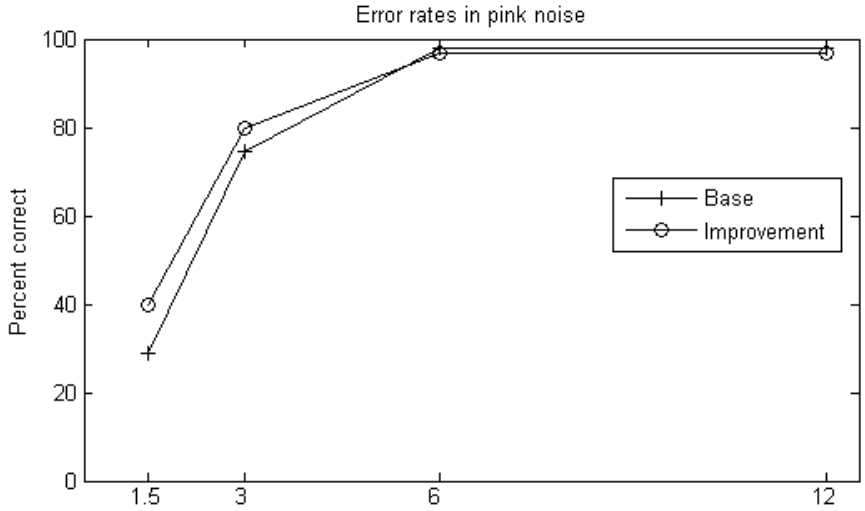

Figure 5.10: Error rates in pink noise

The depression in Figure 5.9 around MFCC 2 is attributed to the noise energy density of pink noise. Although not as apparent in the other higher SNR curves, the SNR curve for

1.5dB clearly shows a preference for the higher order MFCCs.

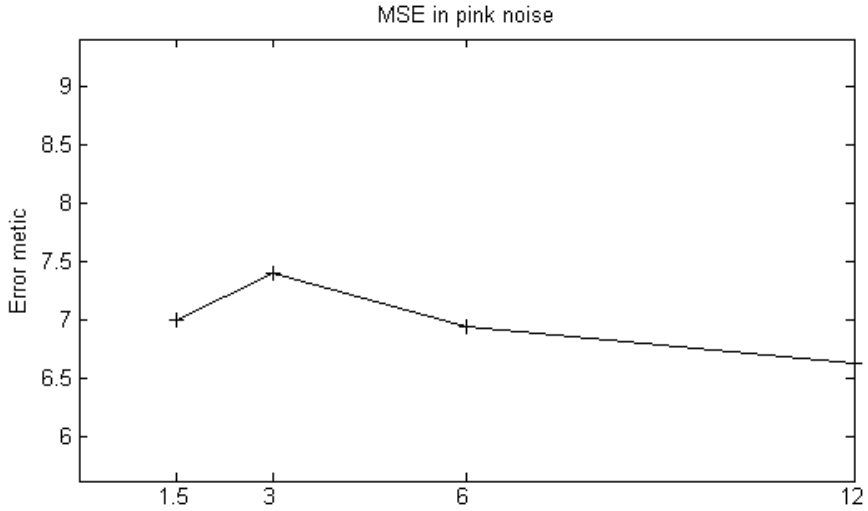

Figure 5.11: Error metric value in pink noise

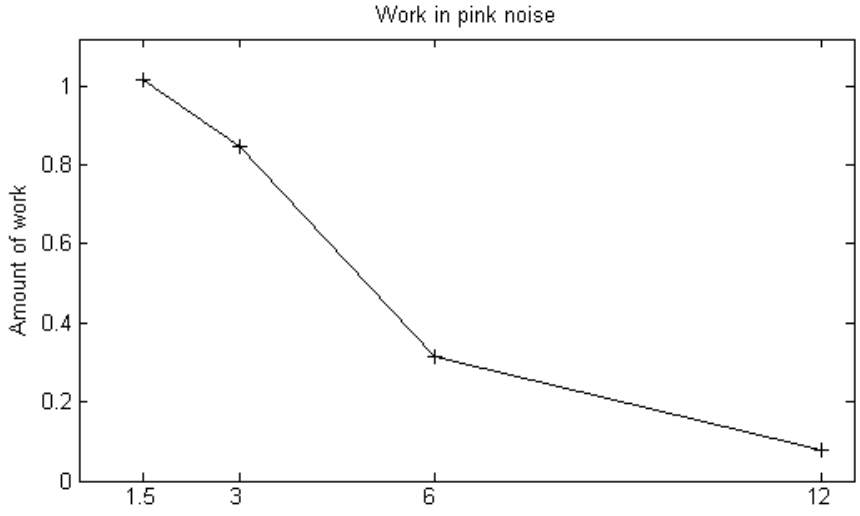

Figure 5.12: Total amount of work in pink noise

Pink noise's metric score curve, as shown in Figure 5.11, appears flatter than of the previous white noises due to the metric's ability to quickly identify a way around the interference of pink noise. The amount of work of the genetic algorithm performed is also much lower for pink noise, as shown in Figure 5.12, than for white noises.

5.1.4 Testing of Brownian Noise

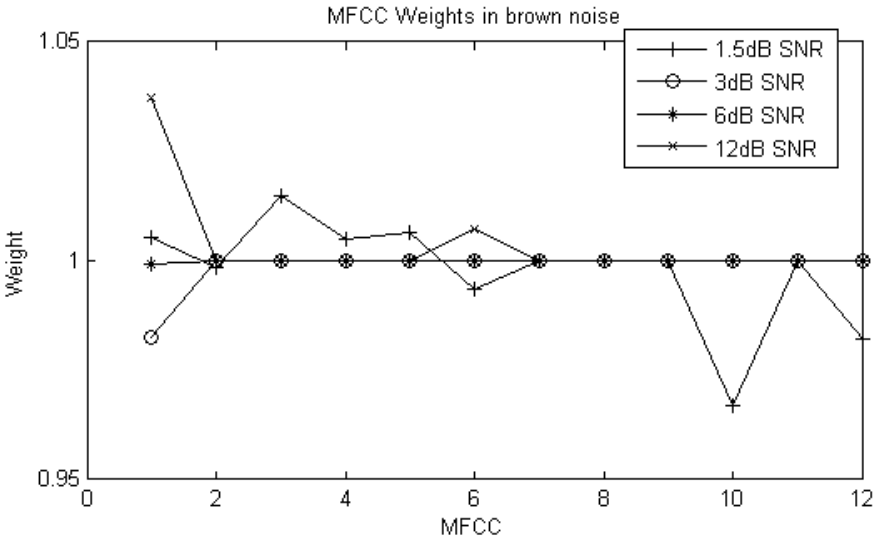

Figure 5.13: MFCC weights in brown noise

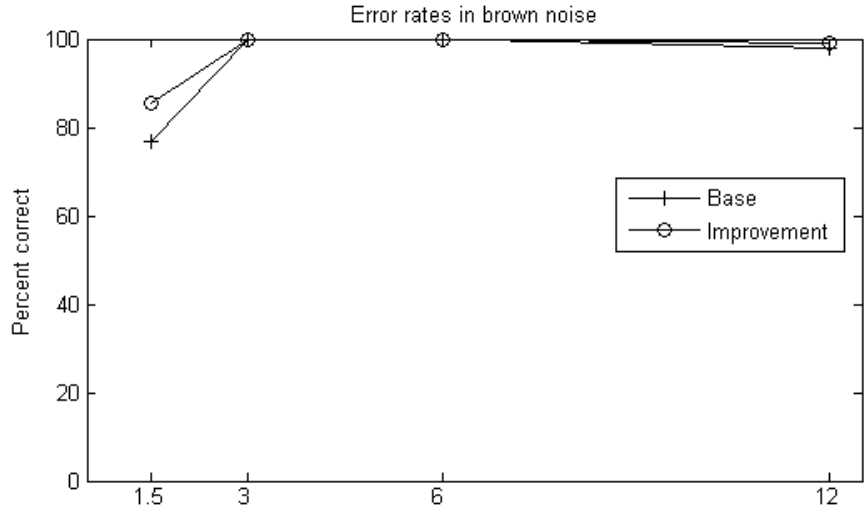

Figure 5.14: Error rates in brown noise

Brownian noise density is more prominent around the lower frequencies explains the lower weights of the MFCCs in Figure 5.13. Due to the noise energy function a substantial increase in error rate was achieved by avoiding the use of the more effected MFCCs, as shown in Figure 5.14.

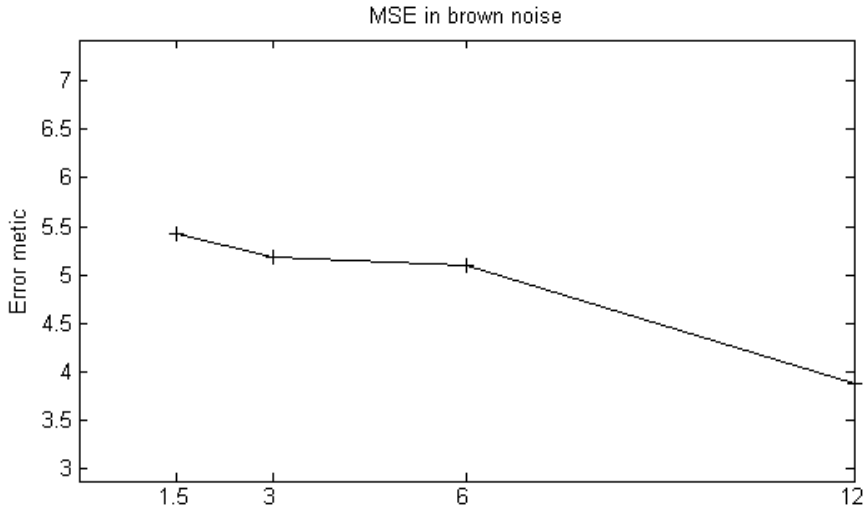

Figure 5.15: Error metric value in brown noise

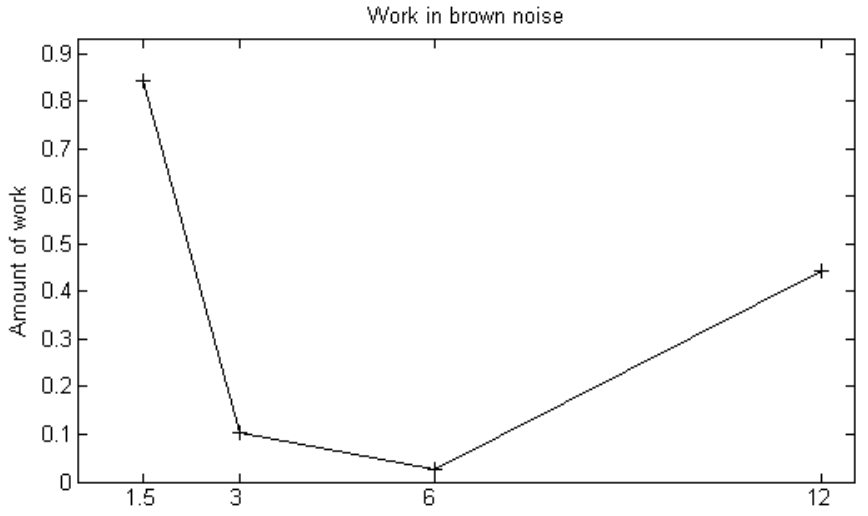

Figure 5.16: Total amount of work in brown noise

The amount of metric error between the clear audio samples and the respective outputs was successfully decreased by a factor of two for the lower SNR tests in Figure 5.14. The slightly better improvement in the higher noise channel can be attributed to the heightened visibility of the noise that helped the metric notice it, as can be inferred from the below unity weight for MFCC 1 in Figure 5.13.

5.2 Channel Models

5.2.1 Testing of AMR

The AMR channel test was conducted by training the GMMs using clear channel audio samples and testing them using varying bitrate AMR channels.

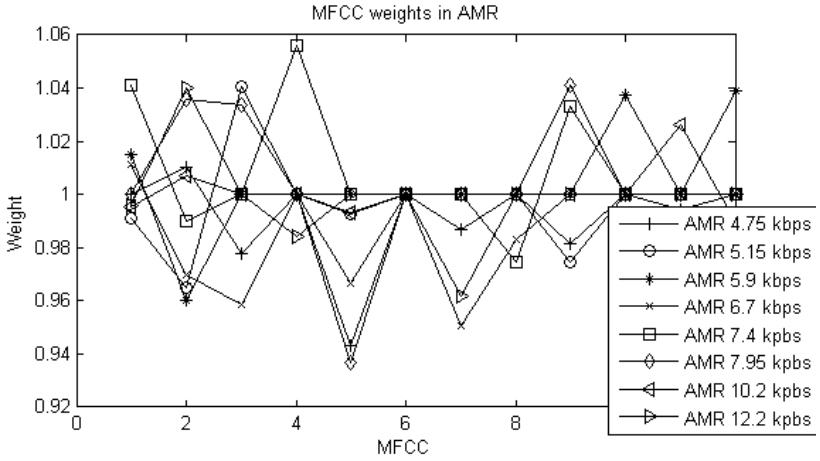

Figure 5.17: MFCC weights in AMR channel

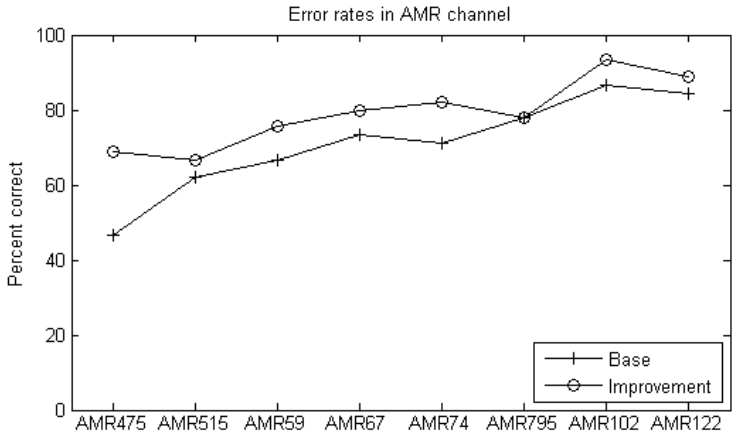

Figure 5.18: Error rates in AMR channel

The MFCC distributions shown in Figure 5.17 differ greatly in interpretation from the noise test results. The greater amount of variation in the lower order MFCCs indicates

AMR encoding more severely effects the deep vowel sounds, which is the phoneme that gives AMR the greatest compression ratio.

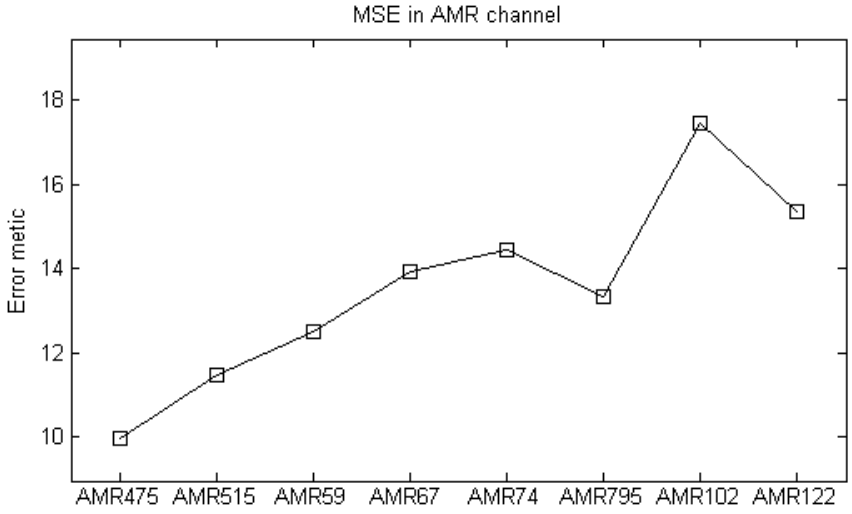

Figure 5.19: Error metric value in AMR channel

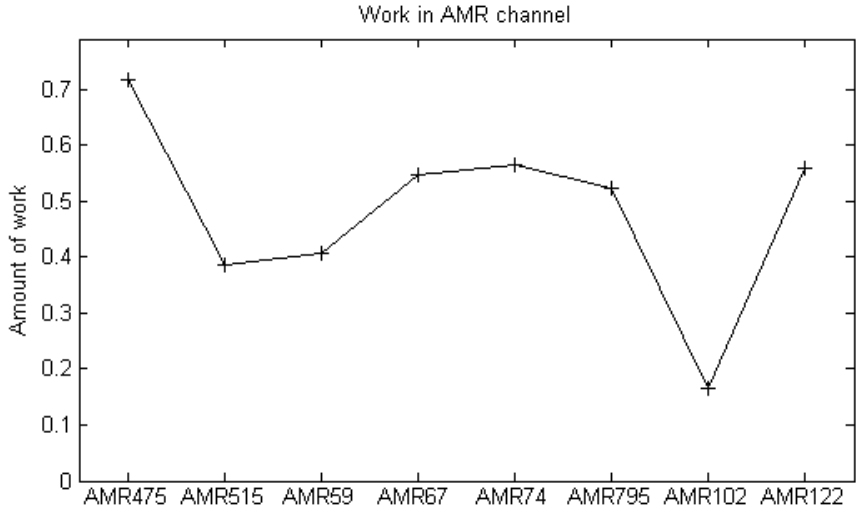

Figure 5.20: Total amount of work in AMR channel

It is interesting to note that the total amount of difference the mutations contributed increased noticeably at half rate AMR (AMR67 through AMR795) in Figure 5.20.

5.2.2 Testing of EVRC

The EVRC channel test was conducted by training the GMMs using clear channel audio samples and testing them using varying bitrate EVRC channels.

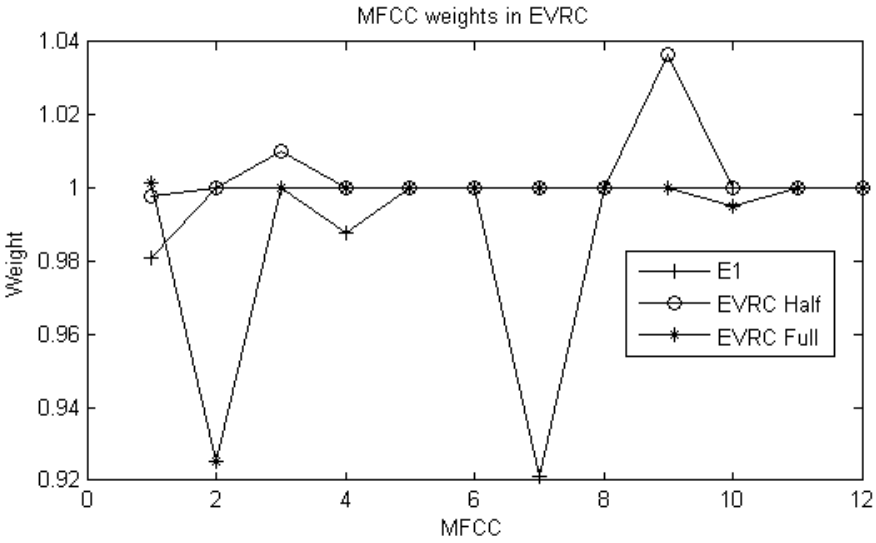

Figure 5.21: MFCC weights in EVRC channel

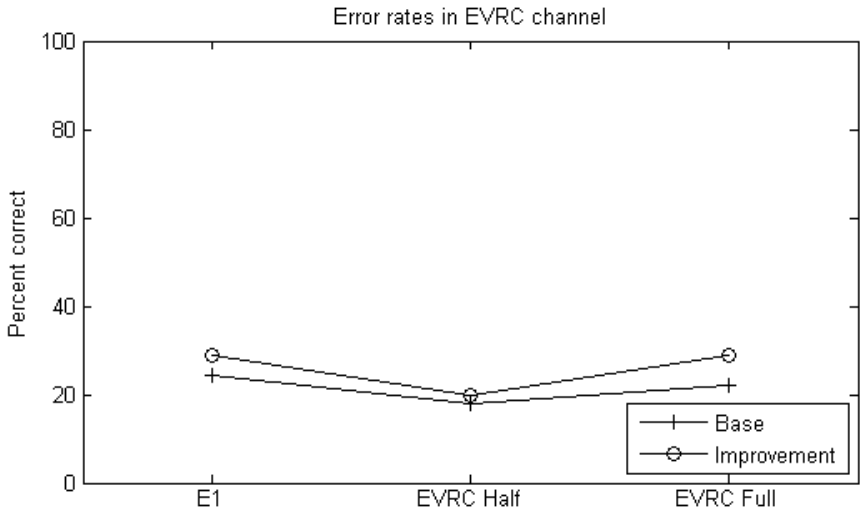

Figure 5.22: Error rates in EVRC channel

EVRC roughly shows the same tendencies as AMR in Figure 5.21. Figure 5.22 shows

EVRC's remarkably poorer performance when compared to AMR. It is thus suggested that for telephone operations EVRC channels be avoided.

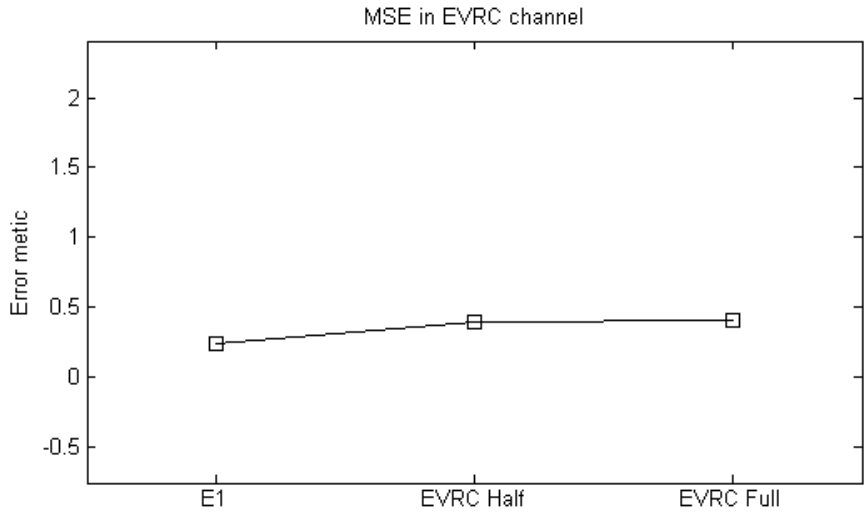

Figure 5.23: Error metric value in EVRC channel

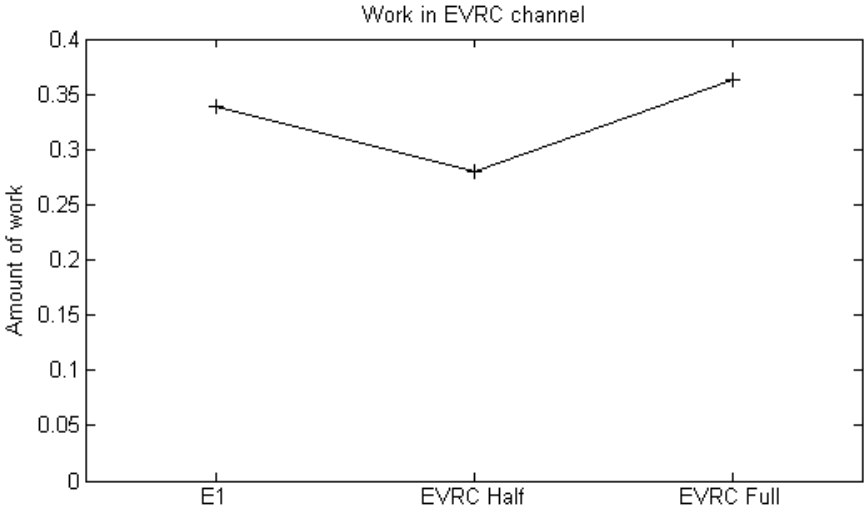

Figure 5.24: Total amount of work in EVRC channel

5.2.3 Testing of Speex

The Speex channel test was conducted by training the GMMs using clear channel audio samples and testing them using varying bitrate Speex channels.

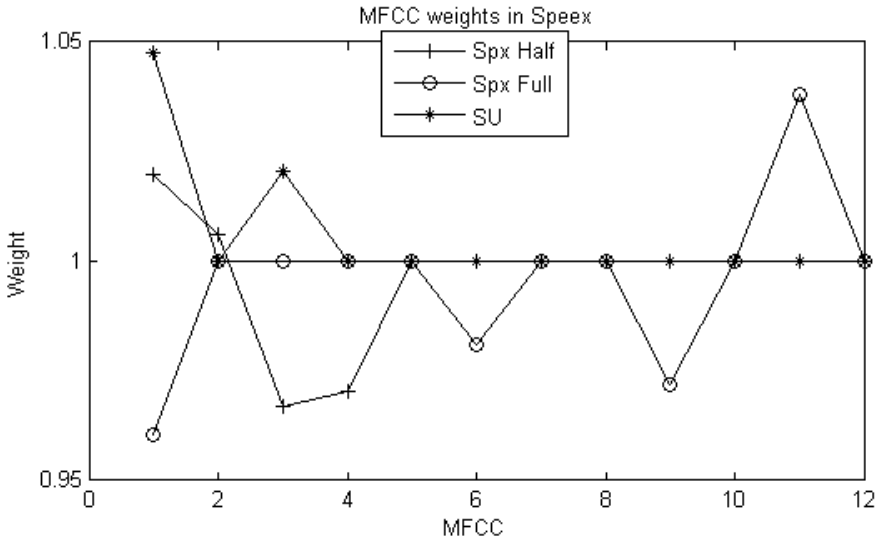

Figure 5.25: MFCC weights in Speex channel

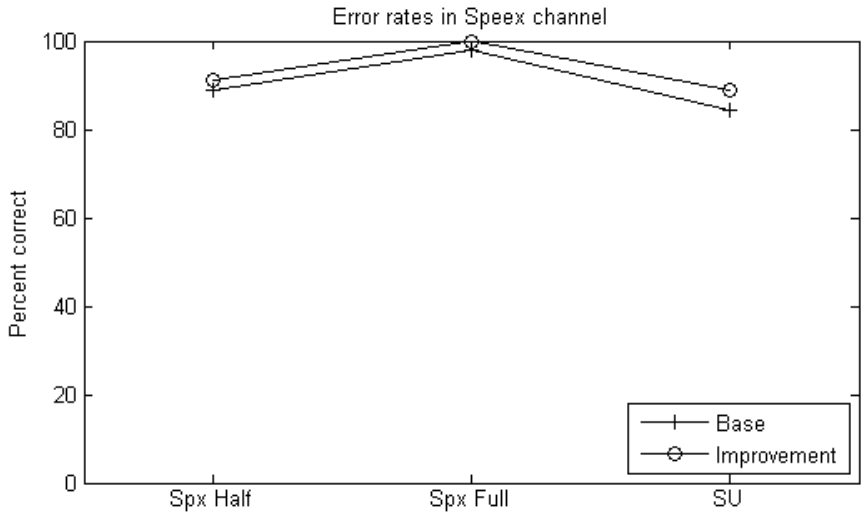

Figure 5.26: Error rates in Speex channel

Speex performs comparatively well compared to AMR at all bitrates. The MFCC distributions in Figure 5.25 correlate perfectly with AMR as well due to the fact that they are

both CELP encoders.

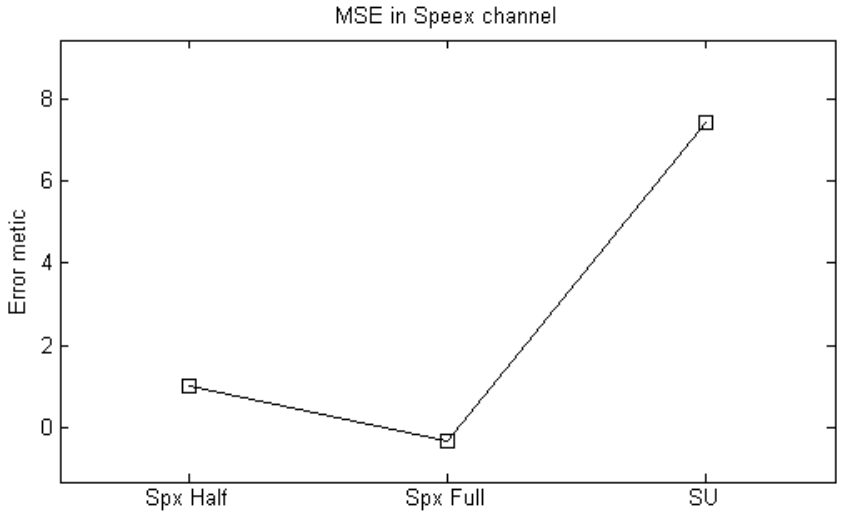

Figure 5.27: Error metric value in Speex channel

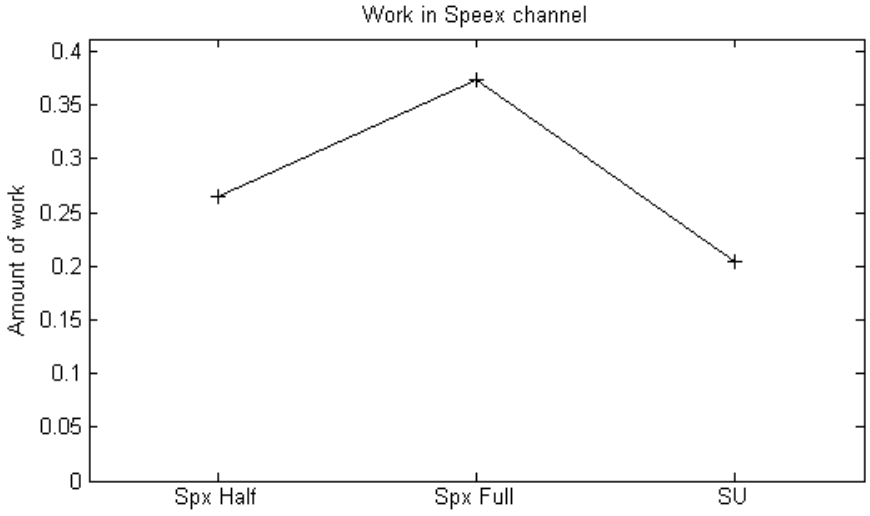

Figure 5.28: Total amount of work in Speex channel

It is interesting to note that Speex works much better with itself than EVRC and AMR.

5.2.4 Testing of AMR full mode

The following test consisted of training the GMMs using audio samples that had passed through AMR in full mode (12.2kbps) and were then tested concurrently against various other full mode codecs.

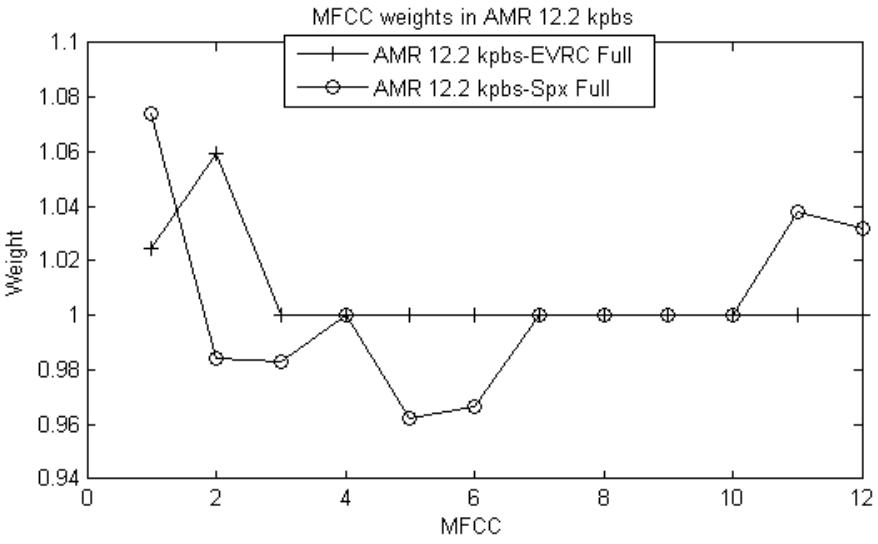

Figure 5.29: MFCC weights in AMR full mode channel

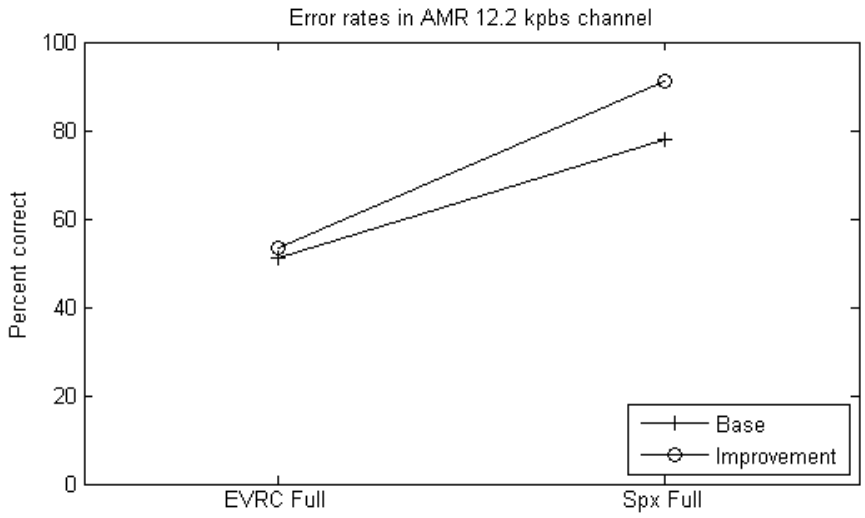

Figure 5.30: Error rates in AMR full mode channel

Figure 5.30 further sustains the conclusion that AMR and Speex work well together.

The improvement in EVRC's performance comes from the fact that AMR and EVRC are both CELP based codecs which makes their spectral features much more similar to one another.

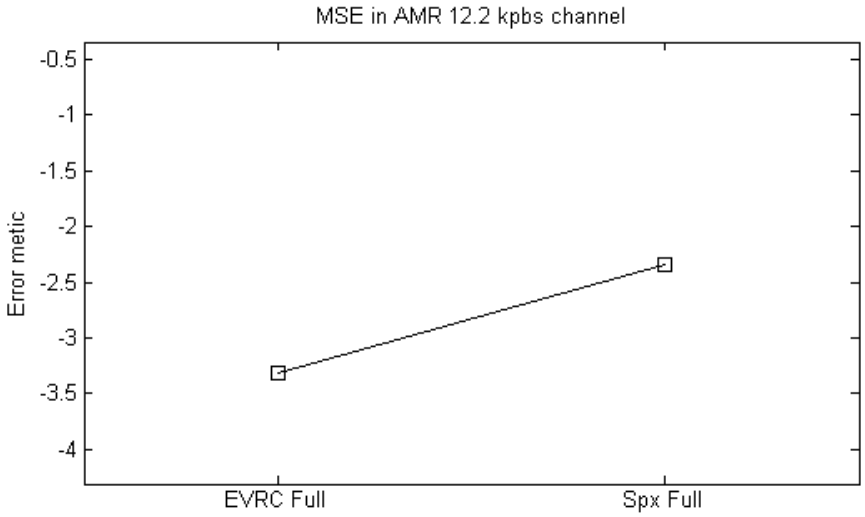

Figure 5.31: Error metric value in AMR full mode channel

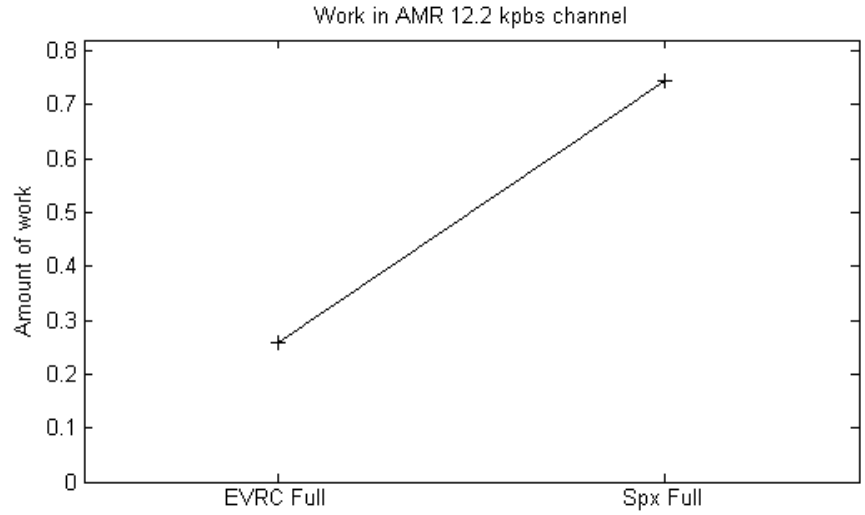

Figure 5.32: Total amount of work in AMR full mode channel

The error rate improvements in Figure 5.30 are greatly correlated with the amount of work exercised by the genetic algorithm, as shown in Figure 5.32. This is the case because

the genetic algorithm was able to find more succesful mutations in the case of the Speex codec than in the EVRC case.

5.2.5 Testing of EVRC full mode

The following test consisted of training the GMMs using audio samples that had passed through EVRC in full mode and were tested against various other full mode codecs.

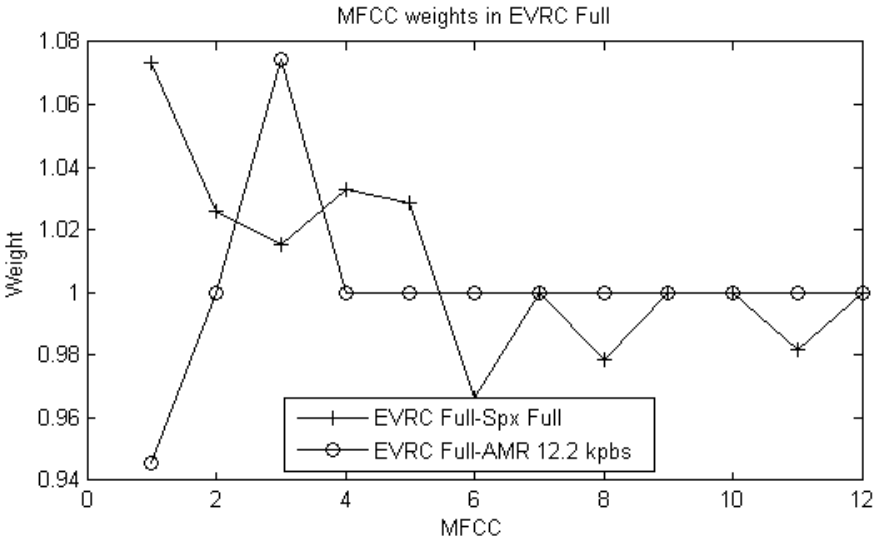

Figure 5.33: MFCC weights in EVRC full mode channel

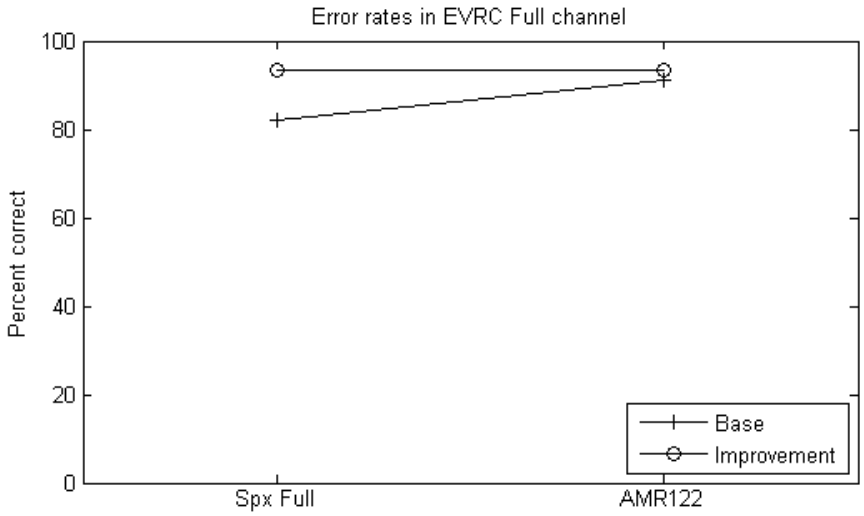

Figure 5.34: Error rates in EVRC full mode channel

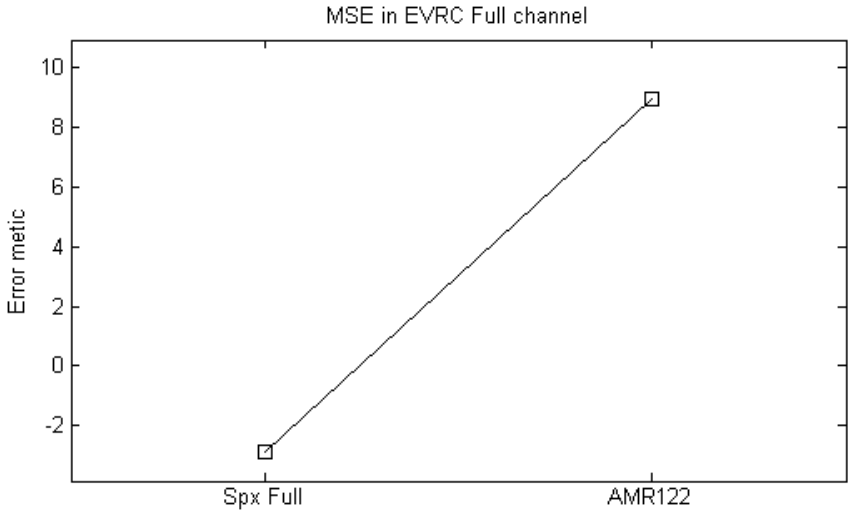

Figure 5.35: Error metric value in EVRC full mode channel

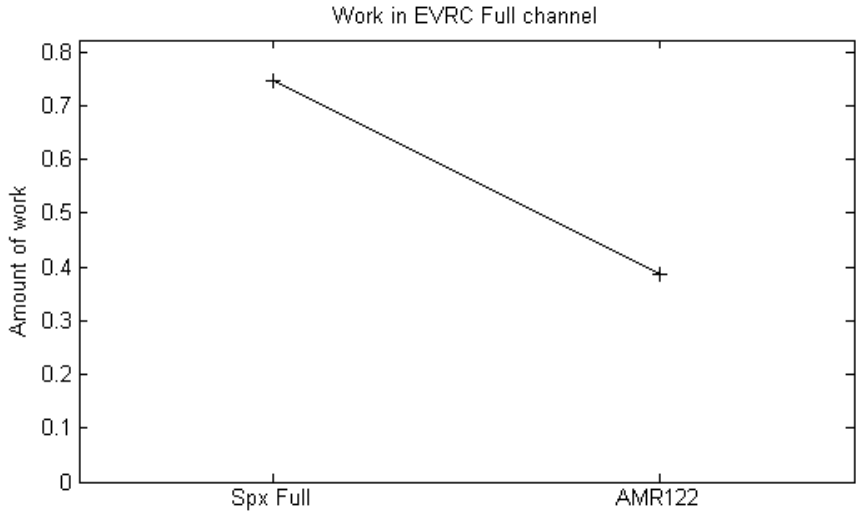

Figure 5.36: Total amount of work in EVRC full mode channel

By comparing Figure 5.30 and Figure 5.34 that EVRC performs better when it is used as the training codec rather than the testing codec. Test audio samples passed through EVRC perform half as well, only being correct nearly half of the time, when compared compared against Speex.

5.2.6 Testing of Speex full mode

The following test consisted of training the GMMs using audio samples that had passed through Speex in full mode and were tested against various other full mode codecs.

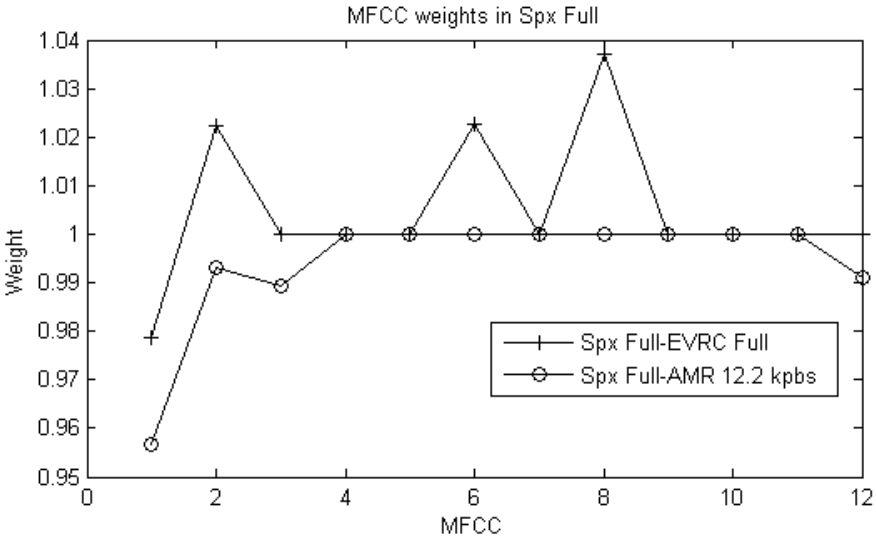

Figure 5.37: MFCC weights in Speex full mode channel

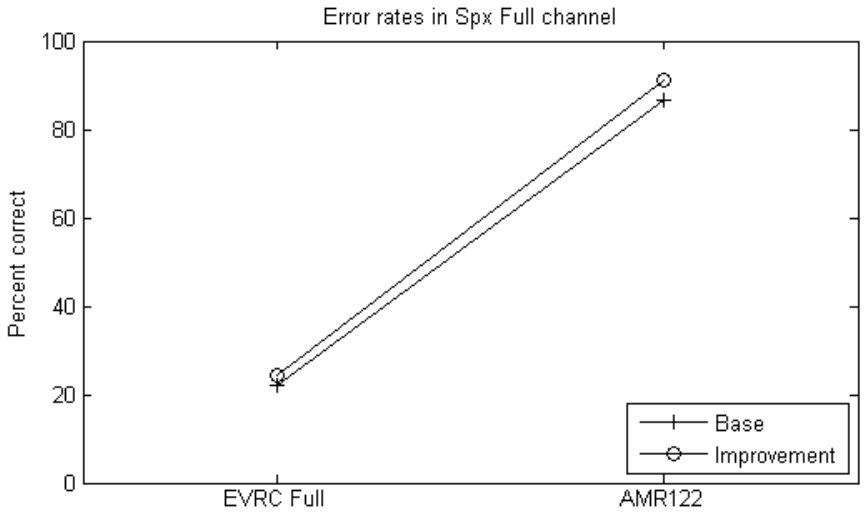

Figure 5.38: Error rates in Speex full mode channel

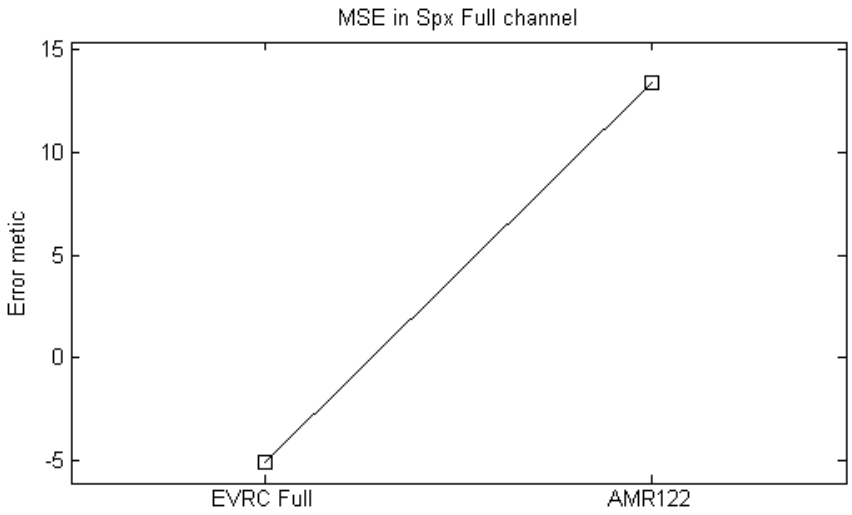

Figure 5.39: Error metric value in Speex full mode channel

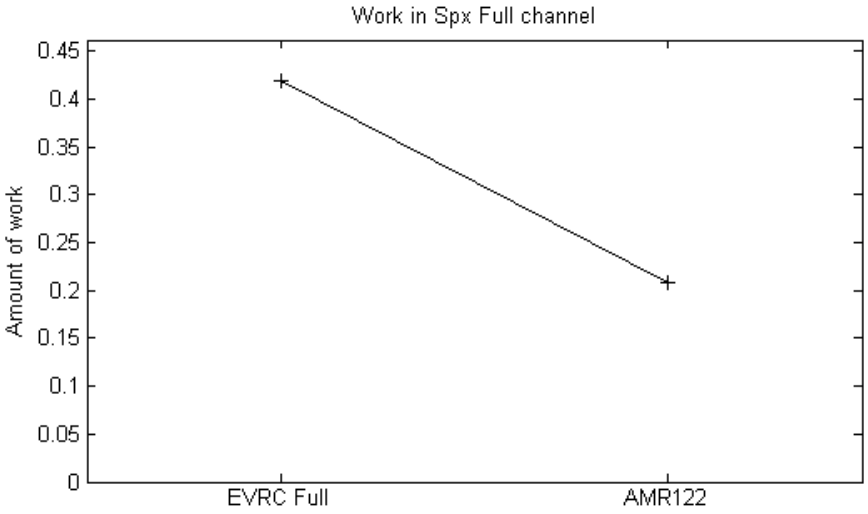

Figure 5.40: Total amount of work in Speex full mode channel

Although EVRC works well with other codecs when it is the training codec, it performs poorly when it is the training codec. This fact is further proved by Figure 5.38. Figure 5.37 additionally shows AMR performing well.

5.2.7 Testing of AMR half mode

The following test consisted of training the GMMs using audio samples that had passed through AMR in half mode and were tested against various other half mode codecs.

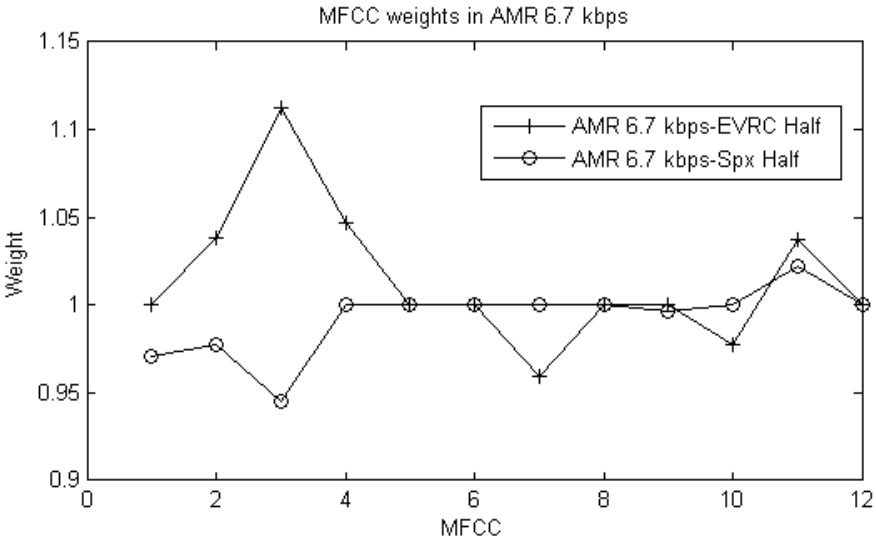

Figure 5.41: MFCC weights in AMR half mode channel

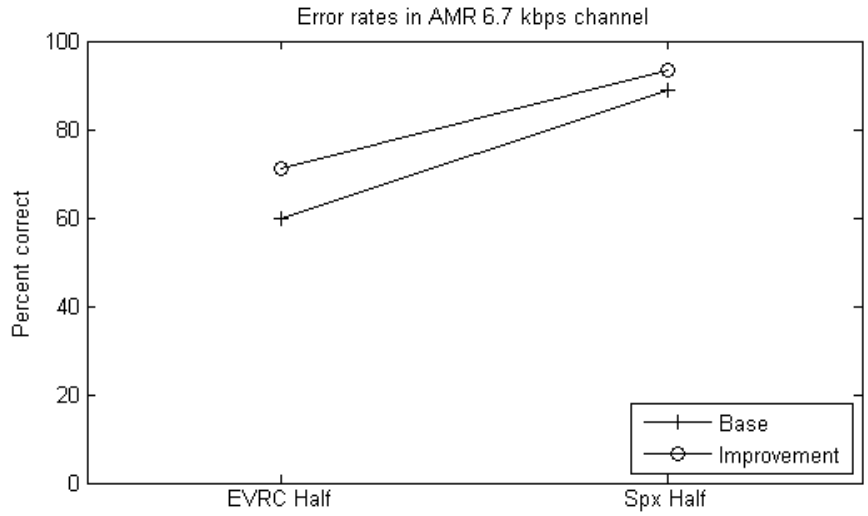

Figure 5.42: Error rates in AMR half mode channel

In Figure 5.42, EVRC is shown to perform remarkably better than when it was tested against GMMs trained using Speex at full rate.

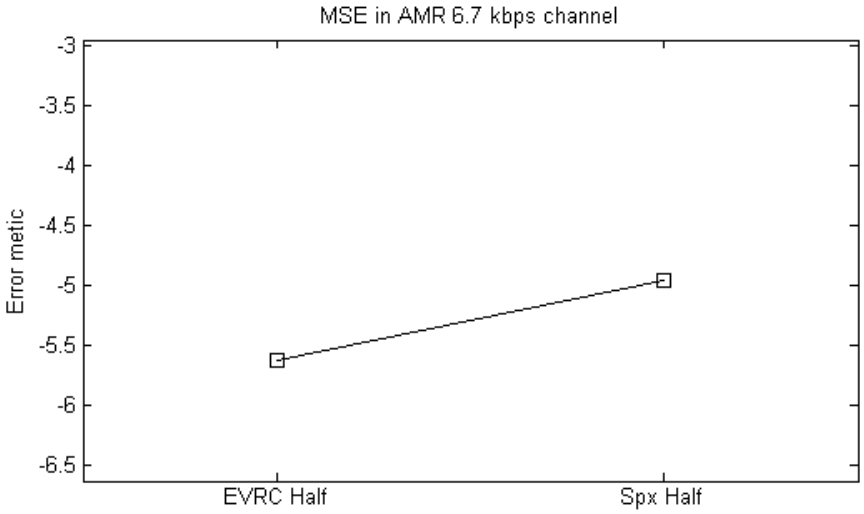

Figure 5.43: Error metric value in AMR half mode channel

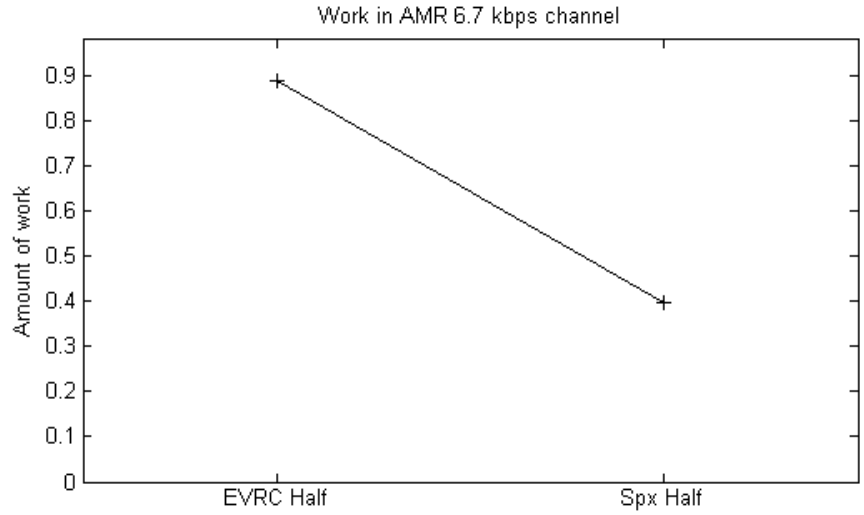

Figure 5.44: Total amount of work in AMR half mode channel

5.2.8 Testing of EVRC half mode

The following test consisted of training the GMMs using audio samples that had passed through EVRC in half mode and were tested against various other half mode codecs.

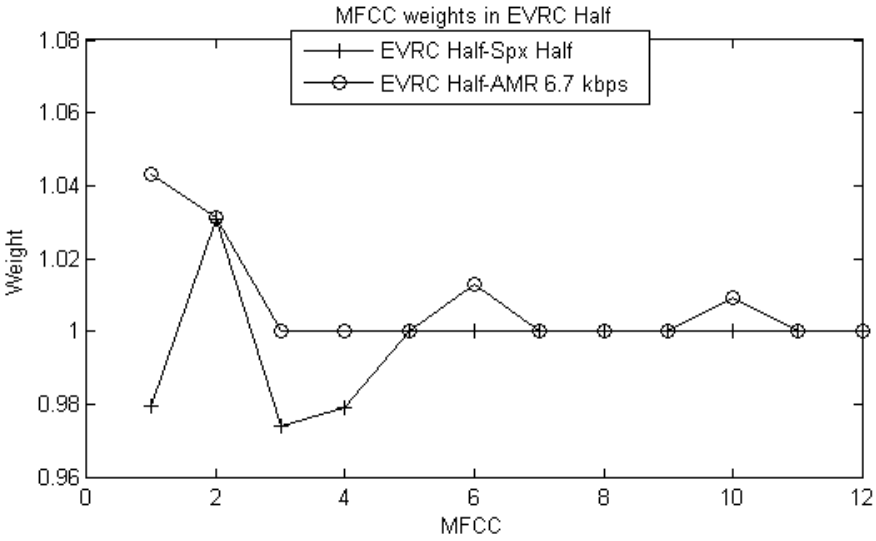

Figure 5.45: MFCC weights in EVRC half mode channel

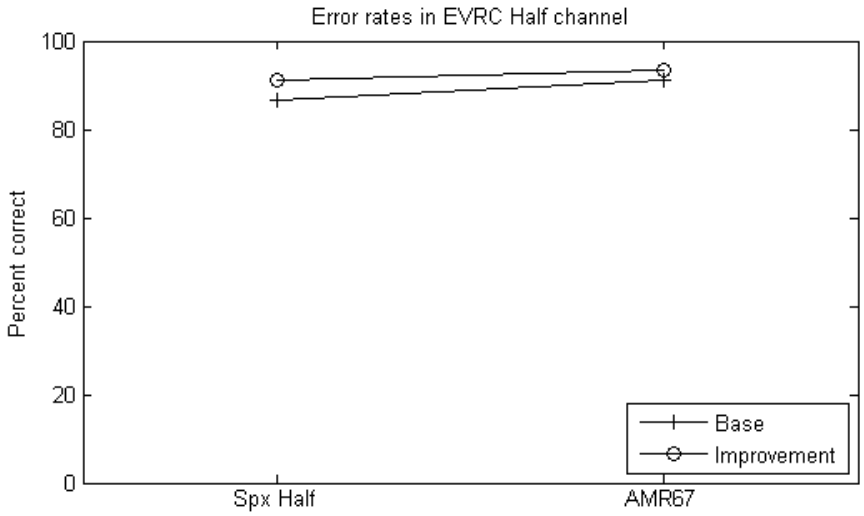

Figure 5.46: Error rates in EVRC half mode channel

The results in Figure 5.46 indicate that EVRC performs better as the training codec.

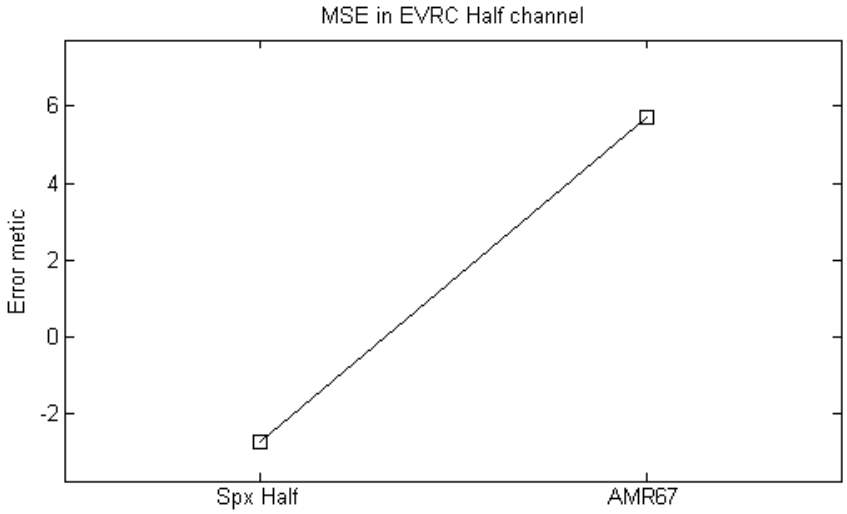

Figure 5.47: Error metric value in EVRC half mode channel

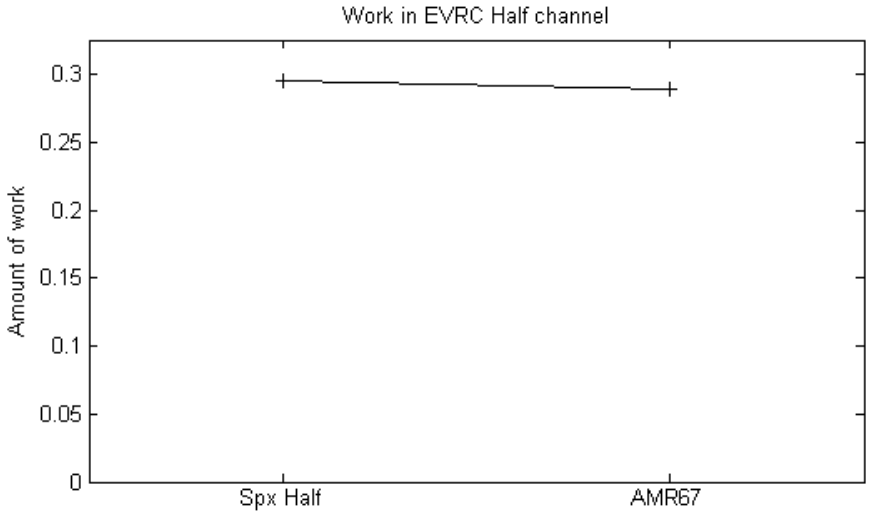

Figure 5.48: Total amount of work in EVRC half mode channel

5.2.9 Testing of Speex half mode

The following test consisted of training the GMMs using audio samples that had passed through Speex in half mode and were tested against various other half mode codecs.

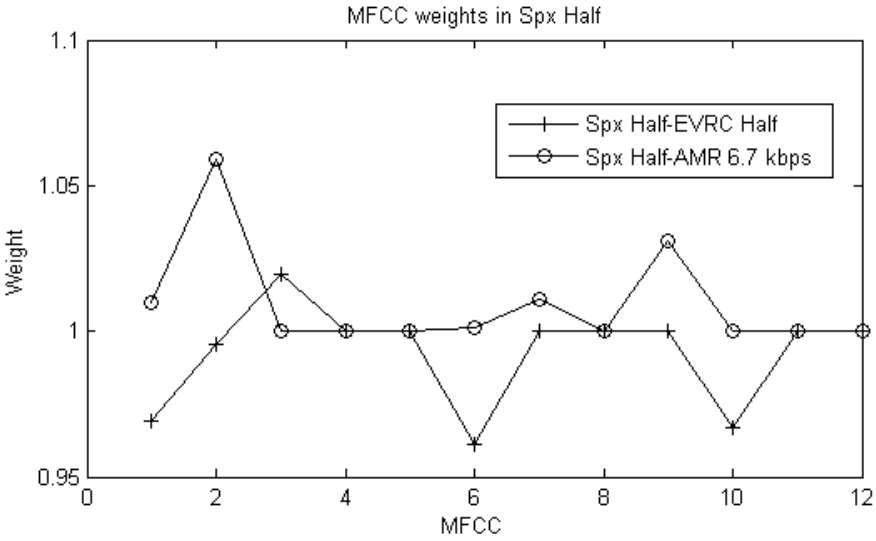

Figure 5.49: MFCC weights in Speex half mode channel

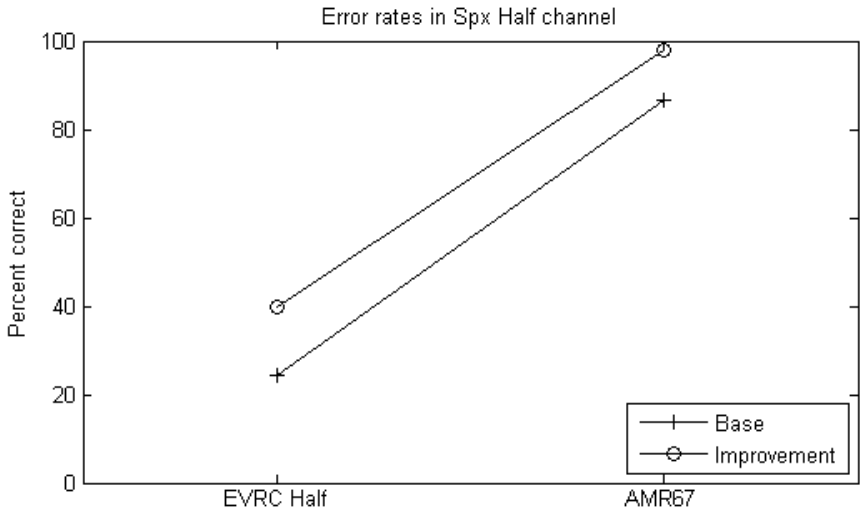

Figure 5.50: Error rates in Speex half mode channel

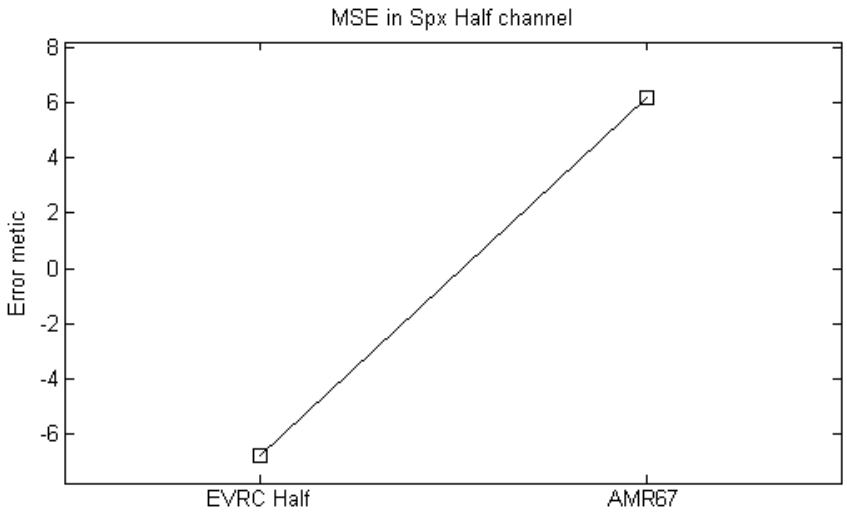

Figure 5.51: Error metric value in Speex half mode channel

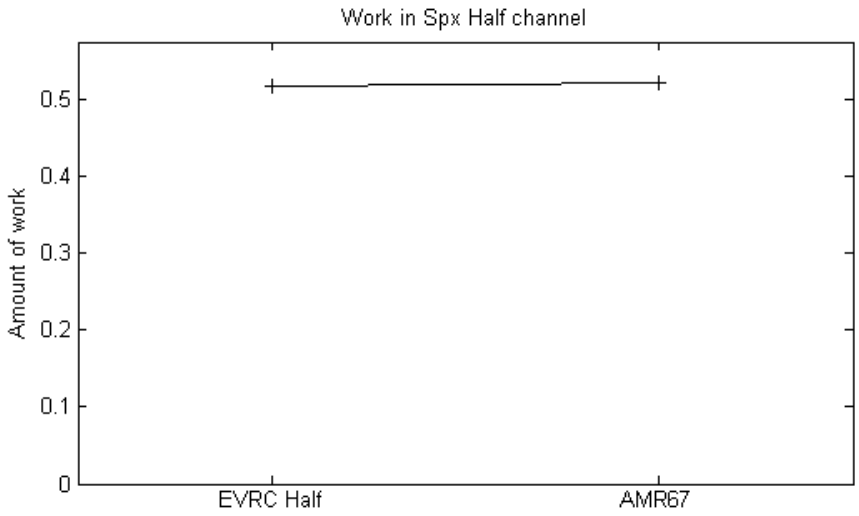

Figure 5.52: Total amount of work in Speex half mode channel

5.3 Remarks

EVRC performs extremely poorly when it is the codec that is used in testing, however this effect was not present when EVRC was used in training. Otherwise no discrepancies between training and testing codecs mismatches were identified. Speex

was observed as performing exceptionally well when its bit rates were mismatched as opposed to other codecs. The MSE rate of each trial was identified as a means of determining the relative amount of improvements that could be brought to each test scenario. Although in a few of the channel noise models the error increased due to the changes of the MFCC weights, all of the codec channel tests showed at least marginal error rate improvements.

Chapter 6

Conclusion

The research strived to implement, analyze, and ultimately improve a modern speaker recognition system that was built upon industry accepted standards. A speaker recognition system utilizing MFCCs for low-level feature extraction and GMMs for classification was verified to be a reliable implementation technique as has been shown in the tests performed in Chapter 4 and Chapter 5.

Through testing it was identified that the stability of the speaker recognition system could be improved by initializing the GMM's statistical parameters with k-means centroids. K-means initializing GMMs was found to produce identical trained statistical models for any given speaker regardless of the selected audio samples. This was very important to ensuring the legitimacy of the thorough tests that followed by removing the GMM's training methodology as a variable in the tests. Additionally, it was found the total number of expectation-maximization algorithm steps decreased when statistical parameters were initialized by k-means centroids.

The NIST-SRE metric was investigated for use in improving a speaker recognition system's performance. It was postulated that error rates would not suffice as a metric in such an application because the speaker NIST-SRE metric would be unable to identify

a decrease in the quality of identifications. As such, a metric that would account for alterations to the quality of identification was proposed. In preliminary testing a Tabu based genetic algorithm was found to encounter more successful mutations and greater improvements to its overall error rate when using the proposed metric instead of the NIST-SRE.

To further the understanding of speaker recognition systems, thorough testing of the speaker recognition was conducted. Testing consisted of noise and channel models that speaker recognition systems encounter in deployment. The effect of various channel impediments were charactized by analyzing results produced by attempting to improve the performance of the speaker recognition system using the proposed metric in combination with the Tabu based genetic algorithm. The results contribute to a better understanding of conditions that influence speaker recognition systems as well as possible remediation techniques that can be invoked by developers.

Although the thesis achieved its goals, many unexplored ideas still remain. Given the proposed metric and the outlined testing methodologies in this thesis, many other improvement techniques in addition to altering MFCC weights can be investigated such as, improving noise filters, and proving the validity of new classifiers.

Bibliography

[1] Xiph.Org, "Speex: A Free Codec for Free Speech," 2011.

[2] S. Vuuren, "Speaker Verification in a Time-Feature Space," *Ph.D. Dissertation*, Oregon Graduate Institute of Science and Technology, 1999.

[3] I. Titze, "Principles of Voice Production," *Prentice Hall*, 1994.

[4] D. A. Reynolds and R. C. Rose, "Robust Text-Independent Speaker Identification Using Gaussian Mixure Speaker Models," *IEEE Transactions on Speech and Audio Processing*, pp. 72–83, Vol. 3, No 1., January 1995.

[5] S. E. Tranter and D. Reynolds, "An overview of automatic speaker diarization system," *IEEE Trans. Speech Audio Processing*, no. 14, pp. 1557–1565, 2006.

[6] D. A. Reynolds, T. F. Quatieri, and R. Dunn, "Speaker verification using adapted Gaussian mixture models," *Proceedings of Digital Signal Processing*, no. 10, pp. 19–41, Vol. 10, 2000.

[7] F. Soong et al, "A vector Quantization Approach to speaker recognition," *Proceedings of ICASSP*, pp. 397–390, 1985.

[8] B. Yegnanarayana and S. Kishor, "AANN: an alternative to GMM for pattern recognition," *Neural Networks*, pp. 459–469, Vol. 15, Issue 3, April 2002.

[9] A. Stolcke, L. Ferrer, S. Kajarekar, E. Shriberg, and A. Venkataraman, "MLLR transforms as features in speaker recognition," *European Conference of Speech Communication Technology*, pp. 2425–2428, 2005.

[10] A. Solomonoff, W. M. Campbell, and I. Boardman, "Advances in channel compesnation for SVM Speaker recognition," *Proceedings of ICASSP (2005)*, 2005.

[11] D. A. Reynolds, "Comparison of background normalization methods for text independent speaker verification," *Eurospeech*, pp. 963–966, 1997.

[12] National Institute of Standards and Technology, *The NIST Year 2010 Speaker Recognition Evaluation Plan*, 2010.

[13] M. Debyeche and A. Amrouche, "Evaluation of Speaker Identification System using GSMEFR speech Data," *Proceedings of the Design and Technology of Integrated Systems in Nanoscale Era (DTIS)*, p. 4, 2010.

[14] M. Phythian, "Effects of speech coding on text-dependent speaker recognition," *TENCON '97*, pp. 137–140, 1997.

[15] D. A. Reynolds, "Large Population Speaker Identenfication Using Clean and Telephone Speech," *IEEE Signal Processing Letters*, pp. 46–48, 1995.

[16] Wang, Jingdong, J. Lee, and C. Zhang, "Kernel Trick Embedded Gaussian Mixture Model," *Proceedings of the International Conference on Algorithmic Learning Theory*, pp. 159–174, Vol. 14, 2003.

[17] P. Ehkan, T. Allen, and S. F. Quigley, "FPGA Implementation for GMM-Based Speaker Identification," *International Journal of Reconfigurable Computing*, no. Article ID 420369, p. 8, Vol 2011, 2011.

[18] V. Wan and S. Renals, "SVMSVM: suppor vector machine speaker verification methodology," *Proceedings of ICASSP*, pp. 221–224, 2003.

[19] E. Jan and J. Flanagan, "Microphone Arrays and Speaker Identification," *IEEE Transactions on Speech and Audio Processing*, pp. 622–629, Vol. 2, No. 4, Oct 1994.

[20] A. Martin, G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki, "The DET curve in assessment of detection task performance," *In Proceedings of the European Conference on Speech Communication and Technology*, pp. 1895–1898, 1997.

[21] D. A. Reynolds, "The effects of handset variability on speaker recognition performance: Experiment on the switchboard corpus," *Proceedings of ICASSP*, pp. 113–116, 1996.

[22] A. Pressman, K. Billings, and T. Morey, "Switching Power Supply Design, 3rd Ed.," *McGraw-Hill*, 2009.

[23] "AMR: 3GPP Specification series," *http://www.3gpp.org/ftp/Specs/html-info/26-series.htm*, 2012.

[24] 3GPP2, "Enhanced Variable Rate Codec, Speech Service Option 3 for Wideband Spread Spectrum Digital Systems," 2004.

[25] "VOICEBOX: Speech Processing Toolbox for MATLAB," *http://www.ee.ia.ac.uk/hp/staff/dmb/voicebox/voicebox.html*, 2012.

[26] A. Sankar and A. Kannan, "Automatic confidence score mapping for adapted speech recognition systems," *Proceedings of ICASSP*, pp. 213–216, 2002.

[27] F. Glover, "Tabu Search - Part 1," *ORSA Journal on Computing*, pp. 190–206, Vol. 1, No. 3, Summer 1989.